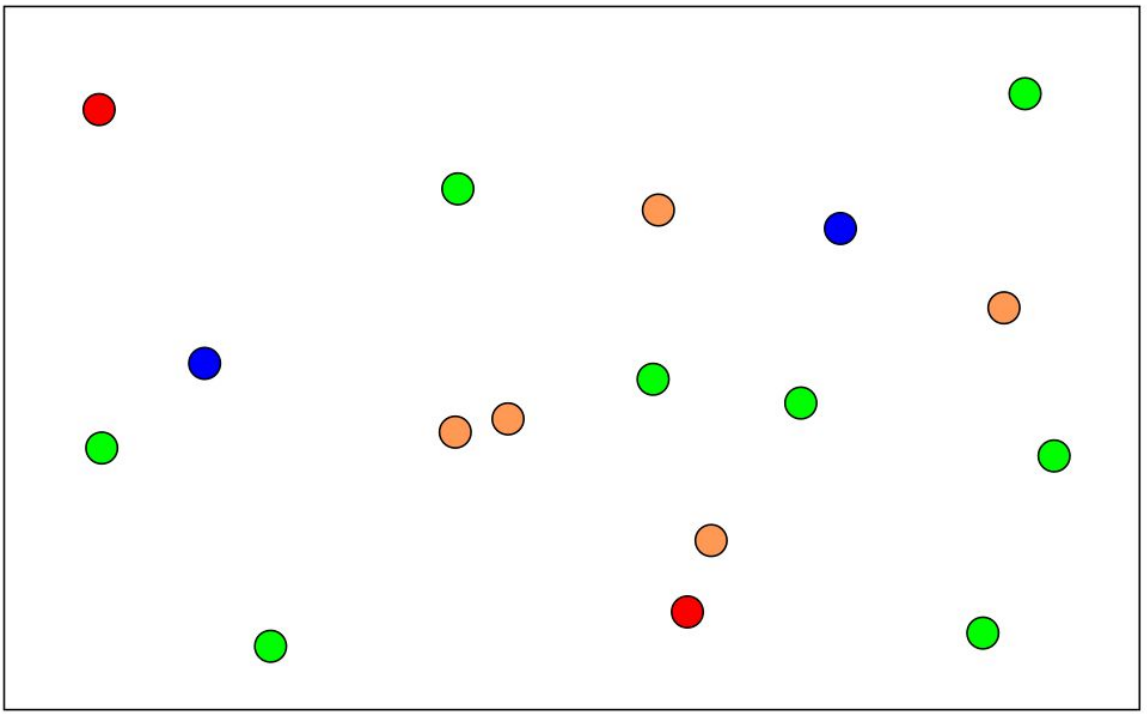
Modelado de la propagación del virus COVID-19

Sistemas de Cómputo Paralelo 2019/20
Practica parte 1: version serie



Ingeniería Informática UPV/EHU
Ingeniería de Computadores Curso 3º
Daniel Ruskov y Mikel Morillo
08/05/2020

ÍNDICE

INTRODUCCIÓN

En la asignatura de Sistemas de Cómputo Paralelo, tal y como su propio nombre indica, se estudian técnicas de programación en la que muchas instrucciones se ejecutan simultáneamente. Se basan en el principio de que los problemas grandes se pueden dividir en partes más pequeñas que pueden resolverse de forma concurrente. En este caso se estudia la programación MPI (Message Passing Interface).

MPI es una especificación para programación de paso de mensajes, que proporciona una librería de funciones para C, C++ o Fortran, diseñada para ser usada en programas que exploten la existencia de múltiples procesadores, y empleada en los programas para comunicar datos entre procesos.

Una vez conociendo la teoría, las funciones de la librería de MPI y su forma de uso, se pretende que, mediante una práctica entregable, se consoliden los conocimientos adquiridos. Dicha práctica consta de dos partes programadas en lenguaje C. Una primera parte o versión para ser ejecutada en serie y que cumple con los requisitos establecidos del enunciado. En la segunda parte, después de tener la primera versión serie funcional, crear a partir de ella la segunda versión, la cual debe ser capaz de modelar el mismo problema mediante una programación paralela MPI.

Este informe contiene información sobre el desarrollo y estructura de la primera versión serie de la práctica.

OBJETIVOS

El objetivo principal de la primera parte de la práctica es crear un sistema capaz simular las interacciones que se producen entre una población de individuos. En este caso, aunque se podría utilizar para diferentes aplicaciones, se va a simular la evolución y los efectos de un virus biológico sobre una población que se desplaza por un escenario compuesto por dos dimensiones.

Debido a que no es posible hacer una representación completamente realista de ningún sistema físico, por la cantidad de variables que hay que controlar, se utilizan modelos que representan ciertas características de los componentes que se quieren modelar. Estos modelos están parametrizados, es decir, dependiendo de los valores con los que se inicializan se comportan de diferente forma. En particular en esta práctica va a modelar el comportamiento del virus COVID-19, obteniendo métricas deseadas en dos ficheros de salida.

Con el desarrollo de esta versión, otro objetivo es consolidar la programación en lenguaje C desarrollando un mejor manejo del mismo y aprender utilidades nuevas. Así mismo, plantear una solución adecuada y manejable, optimizada respecto al tiempo de ejecución y uso de memoria, pero también bien planteada como para ser fácilmente paralelizada en su segunda versión.

DESARROLLO

FICHEROS GENERADOS

En esta primera parte, todo el código generado se presenta unido en un único fichero .c incluyendo las definiciones, la función main y el resto de funciones implementadas.

ESTRUCTURA DEL CÓDIGO

El orden de elementos contenidos en el fichero .c sigue el siguiente orden:

- Autores, fecha y enlaces de interés (comentado).
- Includes para uso de librerías externas (#include <fichero.h>).
- Definición de macros para constantes y funciones MAX(a, b) y MIN(a, b) (#define …).
- Definición de estructuras que modelan características de los componentes (struct T_myStruct {...}).
- Definiciones globales de las variables, punteros y constantes usadas.
- Definición de las funciones implementadas exceptuando la función main. Sirve para resolver las dependencias entre las funciones y como índice.
- Función main.
- Resto de funciones (implementación).
- Código comentado de backup (no necesariamente incluido).

ESTRUCTURAS DEFINIDAS Y ESCENARIO 2D

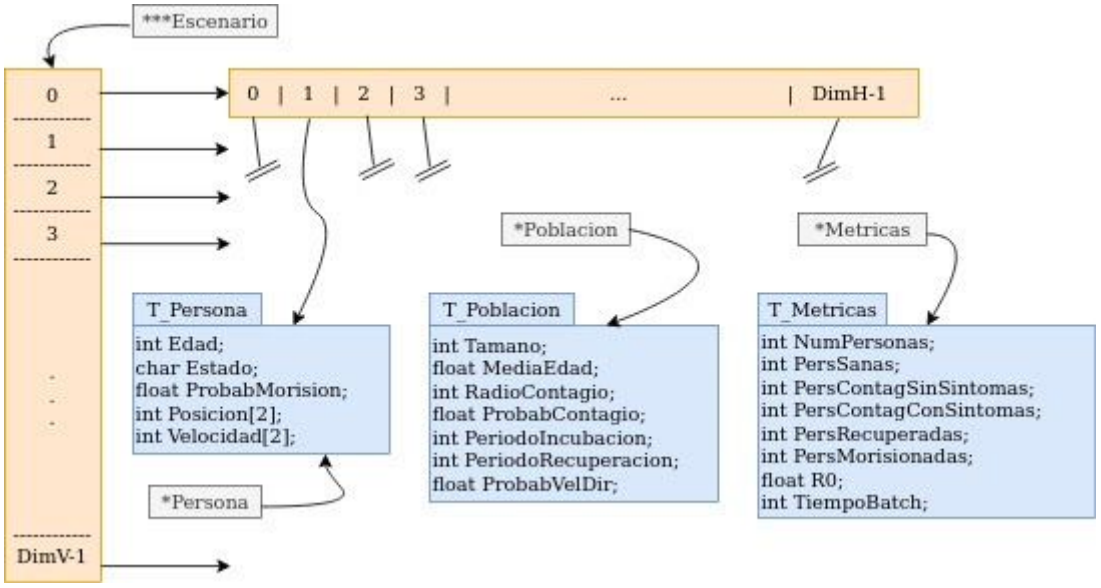
Para modelar el sistema se definen las siguientes tres estructuras:

- **struct T_Persona** // Estructura-Clase que representa una persona en la simulación
  - **int Edad**; // Edad entre 0 y 100
  - **char Estado**; // (0) sano, (1) infectado sin síntomas, (2)infectado con síntomas y (3) recuperado
  - **float ProbabMorision**; // Probabilidad de morir una vez infectado
  - **int Posicion[2]**; // Vector p={px, py} que representa posición de individuo en el escenario
  - **int Velocidad[2]**; // Vector v={vx, vy} que representa dirección y la velocidad de movimiento
- **struct T_Poblacion** // Estructura-Clase que representa la información de la población en la simulación
  - **int Tamano**; // Número máximo de individuos que tiene la población
  - **float MediaEdad**; // Media de edad de los individuos

- **int RadioContagio**; // Contagiados contagian a otros en un radio menor o igual a este parámetro
- **float ProbabContagio**; // Dentro del radio de contagio pueden ser o no contagiados en función de este parámetro
- **int PeriodoIncubacion**; // Tiempo desde contagio hasta muestra de síntomas
- **int PeriodoRecuperacion**; // Tiempo desde muestra de síntomas hasta recuperación
- **float ProbabVelDir**; // Probabilidad de cambio de velocidad y dirección (puede cambiar aleatoriamente)

- **struct T_Metricas** // Estructura-Clase que representa la información recogida en cada unidad de tiempo resultado de la simulación
  - **int NumPersonas**; // Número total de personas en la simulación
  - **int PersSanas**; // Número de personas sanas/no contagiadas
  - **int PersContagSinSintomas**; // Numero de personas contagiadas sin sintomas
  - **int PersContagConSintomas**; // Número de personas contagiadas con síntomas
  - **int PersRecuperadas**; // Número de personas recuperadas
  - **int PersMorisionadas**; // Número de personas fallecidas
  - **float R0**; // Número reproductivo básico: capacidad de contagio o número de personas que es capaz de contagiar un paciente infectado
  - **int TiempoBatch**; // Periodo de tiempo de exportación de métricas a fichero de salida

Para el plano 2D se utiliza una matriz de punteros a objetos de tipo T_Persona. En la siguiente imagen se puede ver la idea de las estructuras y fu forma de utilizarlas junto al escenario de simulación:



struct T_Persona ***Escenario; // Puntero a plano 2D - matriz de punteros a personas
struct T_Poblacion *Poblacion; // Puntero a datos de poblacion
struct T_Metricas *Metricas; // Puntero a datos de métricas

COMPILACIÓN

Para compilar el código y generar el ejecutable es necesario ejecutar el siguiente comando:

> gcc -o *NombreEjecutable NombreFichero.c* -lgsl

Es necesario tener el compilador GCC y las librerías GSL instaladas. El argumento -lgsl hace referencia a la librería GSL de la cual se usa la siguiente función para cálculo de la distribución beta para algunos atributos como la edad:

- *double gsl_ran_beta(const gsl_rng * r, double a, double b)*

EJECUCIÓN. ARGUMENTOS DE ENTRADA

Para la ejecución del programa se necesitan 10 argumentos desde la entrada estándar, siendo el argumento 0 el nombre del programa ejecutable. Los nueve parámetros a continuación deben ser los siguientes respetando el orden:

1. Dimensión vertical del escenario (int).
2. Dimensión horizontal del escenario (int).
3. Número de individuos que va a haber inicialmente en la simulación (int).
4. Duración de la simulación en unidades de tiempo (int).
5. Radio de contagio (int). Distancia a la que puede contagiar una persona enferma. Si un individuo sano se encuentra dentro de es radio, tiene posibilidades de contagiarse.
6. Probabilidad de contagio (float). Probabilidad de que un individuo sano caiga contagiado dentro de un radio de contagio.
7. Periodo de incubación (int). Tiempo que pasa desde el contagio hasta presentar síntomas.
8. Periodo de recuperación (int). Tiempo que pasa desde que se presentan los síntomas hasta recuperarse por completo.
9. Tiempo de batch (int). Periodo de exportación de métricas en unidades de tiempo.

Nota: si se ejecuta con un número diferente de argumentos, se muestra mensaje de error y se señalan los parámetros requeridos. Después se finaliza la ejecución.

SALIDA. EXPORTACIÓN DE MÉTRICAS

Como salida del programa y resultado de la simulación se generan dos ficheros de texto que contienen la información recogida por batches de tiempo:

- **Posiciones_AAAAMMDD-HHMMSS.txt** - consta de cabecera con datos iniciales seguido de las posiciones y estado de cada persona en cada instante de tiempo correspondiente a tiempo de batch. Sigue el siguiente formato:

  ==========

```
Datos de la simulación  2020/05/08 21:48:59
        Número de individuos: 25
        Radio de contagio: 2
        Duración de la simulación: 30
        Periodo de batch para métricas: 2
        Plano de simulación 2D: 6x6
        Estado: (0)Sano, (1)Sin síntomas, (2)Con síntomas y (3)Recuperado
Más datos detallados en el fichero de métricas
========================================================================

t=0
        Pos (0, 0) -> estado 0
        Pos (0, 2) -> estado 0
        Pos (0, 4) -> estado 1
        ...
```

- **Metricas_AAAAMMDD-HHMMSS.txt** - contiene una cabecera con los valores iniciales y una tabla con las métricas obtenidas en cada instante de tiempo correspondiente a tiempo de batch. Su formato es el siguiente:

```
================================================================================================
======
Datos de la simulación    2020/05/08 21:48:59
        Plano de simulación 2D: 6x6
        Número de individuos: 25        Media de edad: 47.00
        Radio de contagio: 2            Probabilidad de contagio: 0.2000
        Periodo de incubación: 8        Periodo de recuperación: 10
        Duración de la simulación: 30   Periodo de batch para métricas: 2
Más datos sobre el movimiento en el fichero de posiciones
================================================================================================
======
Instante t      Sanos   Sin síntomas    Con síntomas    Recuperados     Fallecidos      R0
================================================================================================
======
0               24      1               0               0               0               0.0000
...
```

Nota: Los ficheros se crean automáticamente con nombre que contiene la fecha y hora de creación para que sean únicos. En caso de existir fichero con el mismo nombre, este será reemplazado por el nuevo fichero.

ACLARACIONES

Sigue una lista de aclaraciones sobre dudas que pueden surgir respecto al programa:

- No puede haber más de una persona en el mismo punto del escenario 2D.
- No se puede salir fuera del rango de los límites del plano.
- Se usan funciones de distribución para el cálculo de edades y probabilidades.
    - Distribución ad-hoc - para la probabilidad de morir al estar contagiado.
    - Distribución uniforme 0-1 - para diferentes utilidades.
    - Distribución uniforme rand() para enteros.
    - Distribución beta - para las edades y períodos de incubación y recuperación. Función obtenida de la librería GSL.

- El campo NumPersonas de la estructura Métricas tiene el número de individuos inicial que forman parte de la simulación, pero no disminuye cuando estos fallecen.
- El contador de personas sanas (PersSanas) no incluye el número de personas recuperadas del contador de personas recuperadas (PersRecuperadas) dentro de la estructura Métricas.
- La media de edad (campo MediaEdad de la estructura Población) no se actualiza con los fallecimientos.
- Si se ejecuta el programa con un número de personas extremadamente cercano al número de posiciones posibles, este puede tardar demasiado o no acabar por el while que sirve para encontrar posición libre a la hora de inicializar las personas.
    - *while(Escenario[Persona->Posicion[0]=rand()%DimensionV] [Persona->Posición[1]=rand()%DimensionH] != NULL);*
- Las edades generadas son enteros entre 0 y 100.
- Se procura seguir periodo de incubación de 1 a 14 con media de 5 o 6.
- Se procura seguir periodo de recuperación de 7 a 28 con media de 14.
- Todos los campos de la estructura Población se mantienen constantes durante la ejecución.

INCIDENCIAS

En este apartado se exponen las diferentes incidencias que perduran tras la entrega de la primera parte:

- Incidencias confirmadas:
    - Al ejecutar, tras unos instantes de tiempo las métricas empiezan a mostrar números negativos y la suma no corresponde al número de individuos inicial. No se ha conseguido identificar el error tras varias pasadas de lectura detenida del código y debuggear mediante fprintf(stderr, …).
    - Es posible que no hayamos entendido correctamente el manejo de la función que calcula la distribución beta gsl_ran_beta(...) en lo que respecta a los parámetros que recibe en algunos casos (en los usos dentro del switch durante la simulación). No sabemos ajustar la distribución a nuestra necesidad de forma correcta.
    - Generación de violación de segmento a la hora de cerrar los descriptores de fichero mediante la llamada a la función fclose(...). Para evitarlo, simplemente se igualan a NULL los descriptores de ficheros.
- Incidencias sin confirmar:
    - Dudas si se calcula de forma correcta R0. Actualmente se calcula dividiendo el número de nuevos contagios entre el número de contagios totales. También está la duda de si se contabilizan los contagios sin síntomas para ello (suponiendo que si).

- Dudas sobre si se usan correctamente las distribuciones de probabilidad en lo que respecta a la comparación >= o <=.

CONCLUSIONES

A estas alturas no se pueden obtener muchas conclusiones sin la versión paralela del programa.

En lo que respecta a la productividad del trabajo realizado, podemos concluir que nuestra dinámica es un tanto lenta a la hora de trabajar, pero procurando siempre obtener un resultado correcto o lo más cercano a la solución posible. Cabe añadir que tratamos de dejar todos los aspectos lo más claros posible para su comprensión por terceros.

El desarrollo se ha visto retrasado por motivos de saturación con tareas de otras asignaturas, y al final por errores en la salida del programa (mencionados en el apartado de incidencias) cuyo origen no se ha podido detectar.

Se agradece especialmente el alargamiento del plazo de entregas y la comprensión, apoyo y las explicaciones aportadas por parte del profesor.

MEJORAS

A continuación se listan posibles mejoras respecto al código generado en general o para ser aplicadas en la segunda versión de la práctica, para optimización y adecuaciones:

- Aportar más modularidad.
- Separación en diferentes ficheros (.c y .h) para definiciones y funciones.
- Medición del tiempo de ejecución.
- Uso de buffer para imprimir las cabeceras.
- Uso de buffer para almacenar datos de batches y exportarlos en una sola vez cada cierto tiempo.
- Mejora del formato de ficheros de salida y la información que exportan.
- Evitar condiciones complejas creando funciones auxiliares para claridad del código. Esto puede afectar de forma negativa al tiempo de ejecución por las latencias de salto de las llamadas a funciones.
- Trasladar el código correspondiente a exportación de métricas en función propia.
- Seguimiento de los datos de entrada (formatos, negativos, mayores y/o menores). Condiciones que deben de cumplir.
- Hacer más genericidad a los parámetros que se le pasan a gsl_ran_beta(...).
- Lectura de los argumentos del programa desde un fichero para agilizar la ejecución. Opcionalmente implementar las dos formas de recibir los argumentos.
- Actualizar datos como la media de edad.
- Otras optimizaciones a considerar.

FUTURAS VERSIONES

Como ya se ha mencionado anteriormente en este informe, habrá una segunda versión para la cual se partirá de esta primera solución, utilizando el mismo planteamiento pero de forma paralela, en vez de en serie. Todavía no se ha reflexionado lo suficiente como para plantear una posible solución a estas alturas, pero si se han tenido en cuenta algunas ideas a la hora de desarrollar la primera parte, para que sea más fácil poder paralelizar el código más adelante.

Para realizar la parte paralela del programa, se utilizarán las funciones de MPI. Una vez terminada la siguiente versión, se tomarán mediciones de los tiempos de las dos versiones para comparar y calcular el factor de aceleración y la eficiencia obtenida con la versión paralela. Posteriormente servirá para la toma de conclusiones.

ARCHIVOS RELACIONADOS

Archivos base para la práctica:

- Enunciado de la práctica: https://drive.google.com/open?id=1BVctnzPA6pfx5k3JDsKQ1pnaAvrcqV_0
- Fichero.c con el código correspondiente a la versión serie: https://drive.google.com/open?id=1xwb1shOam7_4bqhWMTin8gHwefxOrVRl

BIBLIOGRAFÍA

Enlaces que pueden resultar de interés y que se han consultado durante el desarrollo:

- Relacionadas con la distribución beta:
  https://en.wikipedia.org/wiki/Beta_distribution
  https://www.gnu.org/software/gsl/doc/html/randist.html