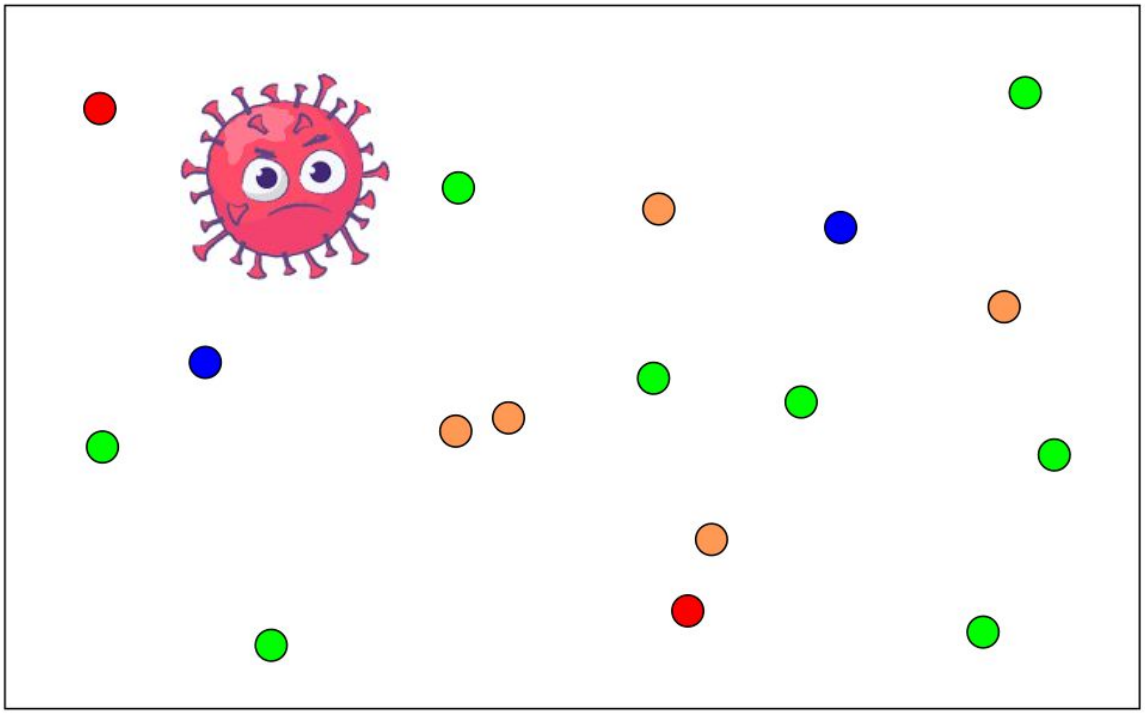
Modelado de la propagación del virus COVID-19

Sistemas de Cómputo Paralelo 2019/20
Práctica final: versiónes serie y paralela



eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

informatika fakultatea
facultad de informática

Ingeniería Informática UPV/EHU
Ingeniería de Computadores Curso 3º
Daniel Ruskov y Mikel Morillo
08/06/2020

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 en el desarrollo y estructura del desarrollo completo 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 la 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.

La segunda versión de la práctica se basa en ajustar el código de la primera versión, de forma que haga uso de funciones MPI para ser finalmente ejecutado el programa

resultado de compilar, y que este haga el mismo trabajo, pero en menos tiempo al dividir las tareas entre diferentes procesos.

Con la segunda parte se pretende aprender y consolidar conceptos básicos de MPI, funciones, estructura del código y manejo y funcionamiento de la programacion y ejecucion con varios procesos formando un cluster.

DESARROLLO

FICHEROS GENERADOS

En la 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. A partir de él y mediante compilador GCC se creó el fichero ejecutable. Ambos están enlazados en el apartado de archivos relacionados de este informe.

En la segunda parte, de la misma forma, todo el código se contiene en un único fichero .c y el ejecutable creado mediante MPICC. Se encuentran referenciados en el apartado de archivos del informe.

ESTRUCTURA DEL CÓDIGO EN AMBAS VERSIONES

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, estructuras 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).

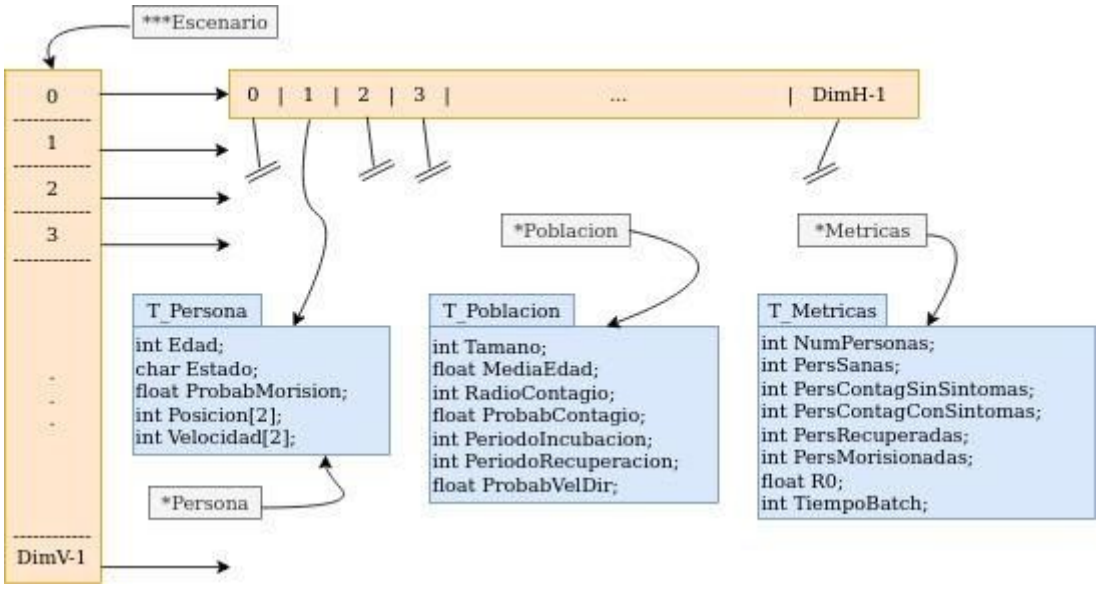
ESTRUCTURAS DEFINIDAS Y ESCENARIO 2D

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 población
struct T_Metricas *Metricas; // Puntero a datos de métricas

COMPILACIÓN

Es necesario tener el compilador GCC y MPICC 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)*

Para compilar el código y generar el ejecutable de la versión serie, es necesario ejecutar el siguiente comando:

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

Para compilar el código y generar el ejecutable de la versión paralela, es necesario ejecutar el siguiente comando:

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

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.

Un ejemplo de la ejecución en ambas versiones sigue el siguiente modelo (aclaración de las razones en el apartado de aclaraciones versión serie):

> ./run.sh "Ejecutable 1000 1000 85000 100 2 0.8 10 10 5"

Al lanzar la ejecución en una terminal, se mostrará feedback por la salida estándar, informando sobre el tiempo de ejecución y que se ha finalizado correctamente. Una ejecución correcta debe tener el siguiente feedback (ejemplo de la versión serie):

```
[scp08@dif-cluster Practica_1]$ ./run.sh "SCP_PracticaSerieMPI_MorilloRuskov 1000 1000 85000 100 2 0.8 10 10 5"
FINALIZACION CORRECTA
  Tiempo inicializacion: 0.03
  Tiempo simulacion: 2.99
  Tiempo total: 3.01
```

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 como se muestra a continuación:

```
[scp08@dif-cluster Practica_1]$  ./run.sh "SCP_PracticaSerieMPI_MorilloRuskov 1000 1000 85000 100 2 0.8 10 10 5"
Error de uso!
USO: ./SCP_PracticaSerie DimV DimH NumIndividuos DuracionSimulac RadioContagio ProbabContagio PeriodoIncubacion
PeriodoRecuperacion TiempoBatch
...
```

SALIDA. EXPORTACIÓN DE MÉTRICAS

Como salida del programa y resultado de la simulación, en ambas versiones 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
...
```

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 VERSIÓN SERIE

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

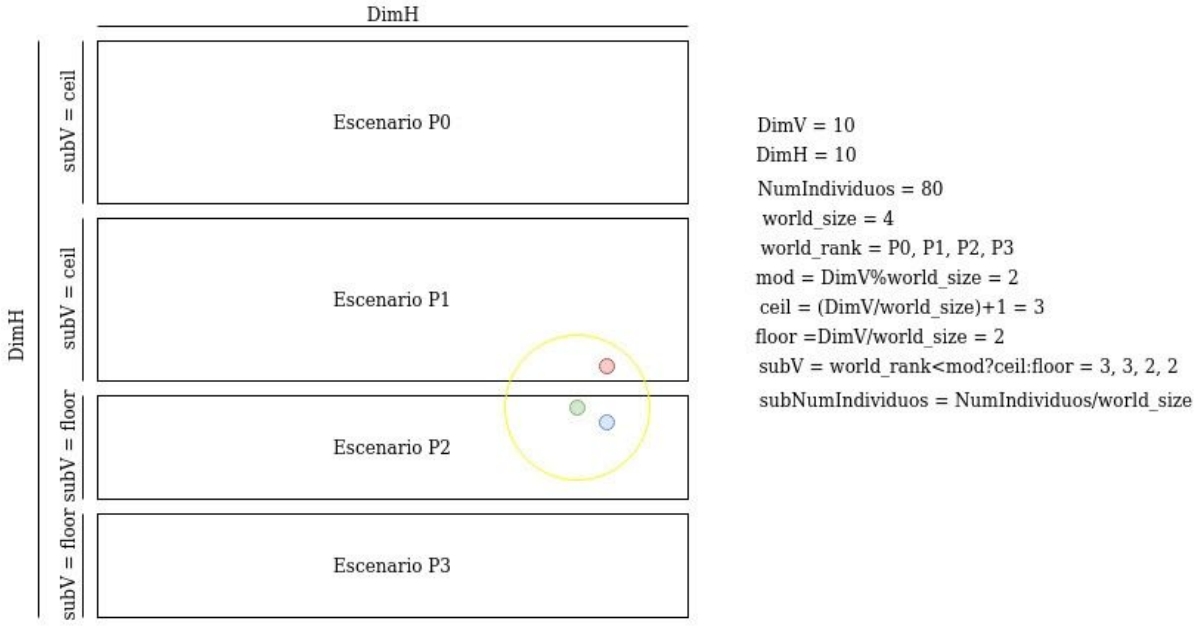
- Hay dos versiones serie. Una de ellas, usada en los ejemplos de ejecución, está hecha con MPI, solo inicialización y finalización MPI, ejecutada en un solo nodo. Esto es para hacer una mejor comparativa de los tiempos ya que se ejecutará en las mismas CPUs que la versión paralela. Ambas versiones serie se adjuntan en el apartado de ficheros relacionados.
- 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.

ACLARACIONES VERSIÓN PARALELA

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

- Se sigue el siguiente reparto entre procesos del plano 2D en la inicialización:



Esto ayuda a repartir el número de filas de la matriz de forma equitativa entre los procesos, y cada proceso inicializa el número de individuos que le puedan corresponder. Cabe mencionar que una mejora puede ser el reparto de número de personas a inicializar según el tamaño del subplano que corresponda.

- Todos los procesos tienen su propia estructura de población y métricas. El root tiene una estructura extra de métricas en la cual se hacen las reducciones de los valores recogidos entre los diferentes nodos antes de un tiempo batch. Una vez hechas las reducciones, todas las métricas se reinician (exceptuando la especial de root que la usa para imprimir la información correspondiente en los ficheros de salida).
- Todos los nodos tienen abiertos los descriptores de fichero de ambos ficheros, pero en el fichero de métricas solo escribe el proceso root.

- En el fichero de posiciones, todos escriben en desorden las posiciones de las personas que tienen en su subescenario, pasada una barrera antes de la cual el proceso root imprime el tiempo de batch y todos se sincronizan. Para imprimir las posiciones reales se hace uso de las variables ceil, mod y floor para calcular la posición vertical.
- Cada proceso tiene una estructura de tipo T_persona auxiliar, que sirve para guardar la información de una persona que migra de un proceso a otro, para posteriormente ser situada en la nueva submatriz del proceso destino.
- Se usa buffer de empaquetado y desempaquetado para los datos de una persona que cambia de proceso. Una vez empaquetado se envía al proceso correspondiente y una vez recibido por el nodo destino se desempaqueta en la estructura de tipo T_persona auxiliar de forma temporal.
- Se añaden al código definiciones y uso de estructuras propias de MPI, también necesarias para el envío y recepción de datos entre procesos.
- En la inicialización el paciente cero se sitúa en el root.

INCIDENCIAS

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

- Generación de violación de segmento a la hora de cerrar los descriptores de fichero mediante la llamada a la función fclose(...) en la versión serie. Para evitarlo, simplemente se igualan a NULL los descriptores de ficheros.
- El ejecutable de la versión paralela no llega a ejecutar el programa completo debido a fallos en tiempo de ejecución:

```
[scp08@dif-cluster Practica_paralelo]$ ./run.sh "exe 100 100 850 50 2 0.8 10 10 2"
[nodo-0-0:21841] *** An error occurred in MPI_File_open
[nodo-0-0:21841] *** reported by process [3057254401,0]
[nodo-0-0:21841] *** on communicator MPI_COMM_WORLD
[nodo-0-0:21841] *** MPI_ERR_INFO: invalid info object
[nodo-0-0:21841] *** MPI_ERRORS_ARE_FATAL (processes in this communicator will now abort,
[nodo-0-0:21841] *** and potentially your MPI job)
[dif-cluster:112871] 3 more processes have sent help message help-mpi-errors.txt / mpi_errors_are_fatal
[dif-cluster:112871] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
```

- Si una persona está en el borde horizontal del subplano de un proceso que hace frontera con otro proceso, se comprueba el radio de contagio perteneciente al proceso que contiene la persona, sin comprobar la parte del radio de contagio que llega al proceso vecino. Esto se debe a que no sabemos resolver bien el problema para explorar el radio de contagio completo y por falta de tiempo. Aquí es donde nos damos cuenta de que la mejor implementación hubiese sido mediante listas ligadas, donde los procesos se reparten el número de individuos y no el plano 2D. Este problema se puede visualizar en la imagen del apartado anterior. La persona no contagiada (punto verde) debería poder contagiarse de la persona contagiada en el proceso vecino (punto rojo), pero no se hace la comprobación del radio de contagio en ese proceso vecino, solo en el rango propio del proceso donde reside la persona, es decir el rango donde se encuentra la persona recuperada (punto azul).

CONCLUSIONES

A estas alturas no hemos obtenido las conclusiones esperadas, ya que la versión paralela no ejecuta debidamente. Se trata de haber obtenido los tiempos de ejecución en ambas versiones con los mismos argumentos de entrada, para hacer una comparativa con datos como el factor de aceleración y la eficiencia conseguida

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.

MEJORAS

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

- Aportar más modularidad.
- Separación en diferentes ficheros (.c y .h) para definiciones y funciones.
- 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.
- No hacer definiciones globales, sino propias de cada proceso.
- Hacer un reparto de trabajo de forma que la simulación quede repartida entre los procesos no root, y el root se ocupa solo de entrada/salida y recogida de métricas.
- Optimizar el uso de memoria. Para ello definir las estructuras y variables propias de cada proceso, sin añadir las que no hacen uso. Relacionado con definir los datos de forma no global.
- Otras optimizaciones a considerar.

Nota: tener en cuenta que siempre se pueden hacer optimizaciones en el tiempo de ejecución con el uso de más memoria y viceversa.

ARCHIVOS RELACIONADOS

- Enunciado de la práctica: https://drive.google.com/open?id=1BVctnzPA6pfx5k3JDsKQ1pnaAvrcqV_0
- Fichero.c con el código correspondiente a la versión serie primera (sin MPI): https://drive.google.com/open?id=1li76m1Rs4a9gUb7bvpc2u3k3F4y9Owp9
- Fichero ejecutable generado a partir del fichero.c version serie sin MPI: https://drive.google.com/open?id=1KVg-KdHvtx5adsIVj_qVog4VoLXUCshj
- Fichero.c con el código de la versión serie MPI con un solo proceso: https://drive.google.com/file/d/1lOsXWFd9xy8E7A3gIczRfhKZd194BEvw/view?usp=sharing
- Fichero ejecutable generado a partir del fichero.c version serie MPI: https://drive.google.com/file/d/1sN-2t1DvbSkMMwUFtEBgCl61X1bZyMrA/view?usp=sharing
- Fichero.c con el código de la versión paralela (falla en ejecución): https://drive.google.com/file/d/17WUpA_S10_A5d5I2K_n0pujBkLRyKgrQ/view?usp=sharing
- Fichero ejecutable versión paralela MPI (falla en ejecución): https://drive.google.com/file/d/12n0UmWRQJZ3MI12nbNht9wWe_So_I_Iu/view?usp=sharing
- Fichero nodes para ejecución MPI: https://drive.google.com/file/d/12GuSUonkCzdjk7Dn3qGzEesdN7HyDU-z/view?usp=sharing
- Fichero run.sh para ejecucion MPI (en el se modifica el número de procesos a usar): https://drive.google.com/file/d/1xnT_VC3bIm7K-OM2W_AuLWP8ftKepze-/view?usp=sharing

BIBLIOGRAFÍA

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

- Relacionados con la distribución beta:
  https://en.wikipedia.org/wiki/Beta_distribution
  https://www.gnu.org/software/gsl/doc/html/randist.html
- Relacionados con programación MPI:
  https://lsi.ugr.es/jmantas/ppr/ayuda/mpi_ayuda.php