

Informe proyecto 1 sistemas operativos

M. Á. González Rodríguez, miguel_gonzalezr@javeriana.edu.co,

I. PDISPERSA

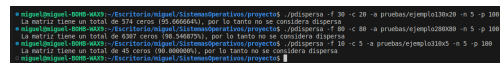
Este código se encarga de cargar la matriz de un archivo en una matriz local. Posteriormente, se dividen las casillas disponibles primero entre filas y luego entre columnas para contar cuántos ceros se encuentran en la matriz. Luego, se compara con el porcentaje dado por parámetro para determinar si es dispersa. A continuación, se presenta el paso a paso:

- 1) datosProceso 'p1' y datosProceso 'p2': Se definen dos estructuras datosProceso, p1 y p2, que almacenan información sobre las filas y columnas iniciales y finales que cada proceso debe procesar en la matriz.
- 2) Se abre un archivo de entrada (archivo) para cargar la matriz desde un archivo.
- 3) Se calcula la variable 'trabajoProcesos', que representa cuántas casillas de la matriz cada proceso debe procesar. Esto se hace dividiendo el número total de elementos en la matriz por el número de procesos y sumando 1.
- 4) Se inicia un bucle while que ejecutará el procesamiento en múltiples procesos hasta que se haya alcanzado el número deseado de procesos ('num_procesos').
- 5) En cada iteración del bucle, se calculan las filas y columnas iniciales y finales ('p1' y 'p2') que cada proceso debe procesar. Estas filas y columnas se calculan de manera que se distribuyan uniformemente entre los procesos.
- 6) Se crea un pipe llamado 'pipe1' para la comunicación entre procesos. Este tubo se utilizará para transmitir una bandera entre los procesos hijo y padre.
- 7) Se crea un primer proceso hijo ('pid1') mediante fork(). Este proceso se encargará de contar la cantidad de ceros en la primera porción de la matriz definida por 'p1'.
- 8) En el proceso hijo ('pid1'), se cuenta la cantidad de ceros en la porción de la matriz definida por 'p1', y si la cantidad supera un umbral (255), se guarda en un archivo ('archivo1').
- 9) Se transmite una bandera al proceso padre a través del tubo 'pipe1' para indicar que el proceso hijo 1 ha terminado.
- 10) Se crea un segundo proceso hijo ('pid2') mediante fork(). Este proceso se encargará de contar la cantidad de ceros en la porción de la matriz definida por 'p2'. Si la suma de ambos procesos hijo supera el umbral, el valor se lee del archivo correspondiente.
- 11) El proceso padre ('pid1' original) espera a que ambos procesos hijos ('pid1' y 'pid2') finalicen su ejecución utilizando waitpid() y luego suma sus resultados parciales.
- 12) Se repite el bucle hasta que se hayan creado y ejecutado suficientes pares de procesos ('pid1' y 'pid2') para

procesar toda la matriz.

- 13) Finalmente, se calcula el porcentaje de ceros en la matriz y se compara con un umbral ('porcentaje'). Se imprime un mensaje indicando si la matriz se considera dispersa o no, según el porcentaje calculado.

Adicionalmente, se adjuntan las ejecuciones con los tres archivos proporcionados en la plataforma, todos con un porcentaje de dispersión del 100%:



```
miguel@igoni:~/000-work$ ./src/bin/miguel/sistemasoperativos/proyecto1/pdispersa -f 20
La matriz tiene un total de 374 ceros (50.000000%), por lo tanto no se considera dispersa
miguel@igoni:~/000-work$ ./src/bin/miguel/sistemasoperativos/proyecto1/pdispersa -f 40
La matriz tiene un total de 6307 ceros (80.568750%), por lo tanto no se considera dispersa
miguel@igoni:~/000-work$ ./src/bin/miguel/sistemasoperativos/proyecto1/pdispersa -f 50
La matriz tiene un total de 45 ceros (59.000000%), por lo tanto no se considera dispersa
miguel@igoni:~/000-work$ ./src/bin/miguel/sistemasoperativos/proyecto1/pdispersa -f 50
```

Fig. 1. Ejecuciones del programa pdispersa [1]

II. HDISPERSA

Este código se encarga de cargar la matriz de un archivo en una matriz local. A diferencia del código anterior, en este caso, los hilos solo se distribuyen por filas, lo que significa que no se pueden crear más hilos que filas disponibles. Se cuenta la cantidad de ceros que se encuentran en la matriz y luego se compara con el porcentaje proporcionado como parámetro para determinar si es dispersa. A continuación, se presenta el paso a paso:

- 1) Se abre el archivo especificado en la variable archivo y se carga su contenido en la matriz 'matrix' utilizando dos bucles for. El primer bucle for itera sobre las filas, y el segundo bucle for itera sobre las columnas de la matriz. La función fscanf se utiliza para leer valores enteros del archivo y almacenarlos en la matriz.
- 2) Después de cargar la matriz desde el archivo, se procede a crear múltiples hilos para procesarla de manera concurrente. El número de hilos creados es igual a 'num_hilos', y cada hilo se encarga de contar la cantidad de unos en un conjunto específico de filas.
- 3) El cálculo de las filas que cada hilo debe procesar se realiza de manera equitativa. Si el número total de filas ('num_filas') es divisible entre 'num_hilos', entonces a cada hilo se le asigna la misma cantidad de filas. Si hay un residuo, es decir, si 'num_filas' no es completamente divisible por 'num_hilos', el residuo se distribuye entre los primeros hilos, asegurando que todas las filas se procesen.
- 4) Cada hilo recibe su rango de filas para procesar, especificado por inicio y final en la estructura 'InfoHilos'. Luego, utiliza dos bucles for anidados para recorrer las filas y columnas dentro de ese rango y busca unos en la matriz. Cada vez que encuentra un uno, incrementa un contador local.

