4. DISEÑO DE ALGORITMOS PARALELOS Y DISTRIBUIDOS

Uno de los ingredientes más importantes para el procesamiento paralelo son sin duda los algoritmos paralelos que tienen un considerable interés en su desarrollo. Dado un problema a resolver en paralelo, el algoritmo paralelo describe como puede resolverse el problema pensando en una determinada arquitectura paralela, mediante la división del problema en subproblemas, comunicando los procesadores y posteriormente uniendo las soluciones parciales para obtener la solución final. Por ejemplo, los algoritmos basados en la estrategia de divide y vencerás usualmente tienen una naturaleza inherentemente paralela.

Existen dos enfoques en el diseño de algoritmos paralelos con respecto al número de procesadores disponibles. El primero es el diseño de un algoritmo en el cual el número de procesadores utilizados por el algoritmo es un parámetro de entrada, lo que hace que el número de procesadores no dependa del tamaño de entrada del problema.

El segundo enfoque permite que el número de procesadores utilizados por el algoritmo paralelo crezca con el tamaño de la entrada, de tal forma que el número de procesadores no es un parámetro de entrada, pero si una función del tamaño de entrada del problema. Utilizando el esquema “división de labor”, un algoritmo diseñado con el segundo enfoque puede siempre ser convertido a un algoritmo del primer enfoque.

4.1. Patrones de Comunicación Paralela

En relación a los patrones de comunicación que se establecen entre los procesos de una aplicación paralela y distribuida existen varias posibilidades de interacción: los farms (o granjas de procesos), los pipes (o cauces), los trees (o árboles), los cubes (o cubos o hipercubos), los grids (o mallas), las matrices de procesos, etc. De tal forma que saber qué patrón se adapta mejor a una aplicación determinada constituye una decisión importante de diseño que no puede ser totalmente automatizada. Una vez identificado el comportamiento paralelo de la aplicación en desarrollo, el segundo paso consiste en elaborar un bosquejo gráfico de su representación, con los detalles concretos de ésta: número de nodos, canales de conexión entre estos, etc., como un documento previo al diseño detallado en el que se detallará posteriormente el procesamiento paralelo del sistema objetivo.

Cuando ya se tiene concretizado el modelo que define un patrón paralelo específico, digamos, por ejemplo, un tree, o alguno de los anteriormente mencionados, el paso siguiente será realizar su definición sintáctica y semántica. Traduciéndose finalmente la definición sintáctica a un programa en el entorno de programación más adecuado

para su implementación paralela ha de verificarse que la semántica resultante sea la correcta; es conveniente probarlo con varios ejemplos distintos para demostrar su genericidad así como observar el rendimiento de las aplicaciones que lo incluyan como un componente software.

Los patrones paralelos que veremos a continuación son: el pipeline, el farm y el tree bajo la técnica de divide y vencerás, ya que se trata de un conjunto significativo y reutilizable de patrones utilizados en múltiples aplicaciones y algoritmos con diferentes propósitos, en diferentes áreas y con diferentes aplicaciones prácticas, según la literatura que hay sobre el tema. La configuración de cada uno de estos patrones es la siguiente:

- el pipeline está compuesto por un conjunto de estados conectados uno detras de otro, la información sigue un flujo de uno a otro;
- el farm se compone de un conjunto de procesos trabajadores y un proceso controlador; los trabajadores se ejecutan en paralelo hasta alcanzar un objetivo común y el controlador es el encargado de distribuir el trabajo y de controlar el progreso del cómputo global;
- el tree, hace que la información fluya desde la raíz hacia las hojas o viceversa, ejecutándose los nodos que se encuentran en el mismo nivel en el árbol en paralelo; la técnica de diseño de algoritmos denominada Divide y Vencerás utiliza este patrón para obtener la solución del problema inicial.

El Patrón de Comunicación PIPELINE

La técnica del procesamiento paralelo del pipeline es aplicable a la resolución de un amplio rango de problemas que son parcialmente secuenciales en su naturaleza. Utilizando la técnica del Pipeline, la idea es dividir el problema en una serie de tareas que tienen que ser completadas una después de otra. En un pipeline cada tarea puede ser ejecutada por un proceso, un thread o un procesador independiente [ROB], (figura 18):

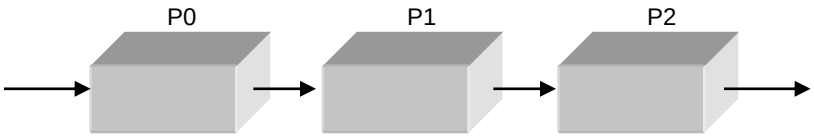

Fig. 18. PipeLine

Algunas veces a los procesos del pipeline se les llama etapas (stages) del pipeline [SEL99]. Cada etapa puede contribuir a la solución del problema total y puede pasar la información necesaria a la siguiente etapa del pipeline. Este tipo de paralelismo es visto muchas veces como una forma de descomposición funcional, ya que el problema es dividido en funciones separadas que pueden ser ejecutadas individual e independientemente; pero con esta técnica las funciones se ejecutan en sucesión. Un algoritmo que resuelva un determinado problema puede ser entonces formulado como un pipeline si se puede dividir en una serie de funciones que podrían ser ejecutadas por las etapas del pipe [ROB]. Supongamos, por ejemplo, que queremos ordenar un conjunto de datos en desorden de mayor a menor (es decir, en orden descendente), pero se tiene ya implementado un algoritmo de ordenación de menor a mayor o en orden ascendente; si se utilizase este algoritmo de ordenación, sería necesario invertir la secuencia de los datos ya ordenados, lo que se puede llevar a cabo añadiendo un stage extra al pipeline con una función asignada que llevaría a cabo el proceso específico (figura 19).

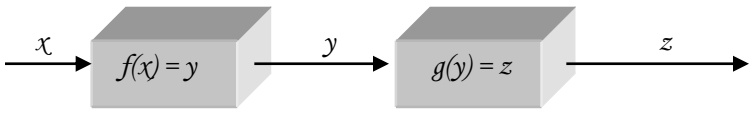

Fig. 19. Pipeline con descomposición funcional

La interpretación de los elementos de la figura anterior es como sigue:

- x representa al conjunto de datos inicial que supondremos está en desorden;
- f(x) representa la función “ordena” que recibe como entrada el conjunto de datos a ordenar y proporciona como salida la ordenación en orden ascendente (de menor a mayor) del conjunto de datos que le llegan;
- y representa entonces la salida de la función f(x), es decir, los datos ordenados;
- g(y) representa la función “invierte” que recibe el resultado de la función “ordena” para dar como salida el conjunto de datos previamente ordenados pero invertidos en su secuencia para tener un orden descendente (de mayor a menor);
- z es precisamente la salida de la función g(y) en la última etapa del pipeline.

Suponiendo que el conjunto de datos con los que se trabaja en este ejemplo son números enteros, la secuencia de resultados dentro del pipeline sería la mostrada en la figura 20.

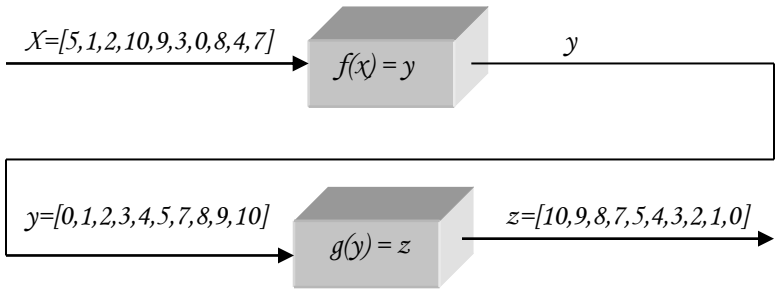

Fig. 20. Secuencia de resultados en un Pipeline

Así, si un problema que puede ser dividido en una serie de tareas secuenciales, el enfoque del pipeline puede proporcionar incremento en la velocidad de ejecución de los siguientes tres tipos de cálculos:

1. Cuando más de una instancia del problema completo puede ser ejecutada en paralelo;
2. bien una serie de datos pueden ser procesados y cada uno de estos se utiliza en múltiples operaciones;
3. bien si la información que demanda el siguiente proceso para iniciar su cálculo se pasa después de que el proceso actual haya completado todas sus operaciones internas.

Con esta técnica muchos de los problemas computacionales que se llevan a cabo de forma secuencial pueden ser fácilmente paralelizados como un pipeline. Ejemplos de estos problemas son:

- la suma de números,
- el ordenamiento de números,
- la generación de números primos,
- el solucionar un sistema de ecuaciones lineales.

El Patrón de Comunicación FARM

El patrón paralelo de interacción Farm está formado de un conjunto de procesos independientes entre sí, llamados procesos trabajadores, y un proceso que los controla, llamado el proceso controlador [ROB], [SEL99]. Los procesos trabajadores se ejecutan en paralelo hasta alcanzar un objetivo común para todos ellos. El proceso controlador es el encargado de distribuir el trabajo y de controlar el progreso del farm hasta hallar la solución del problema que se pretende resolver en la aplicación que utilice este patrón. La figura 21 muestra el modelo del Farm.

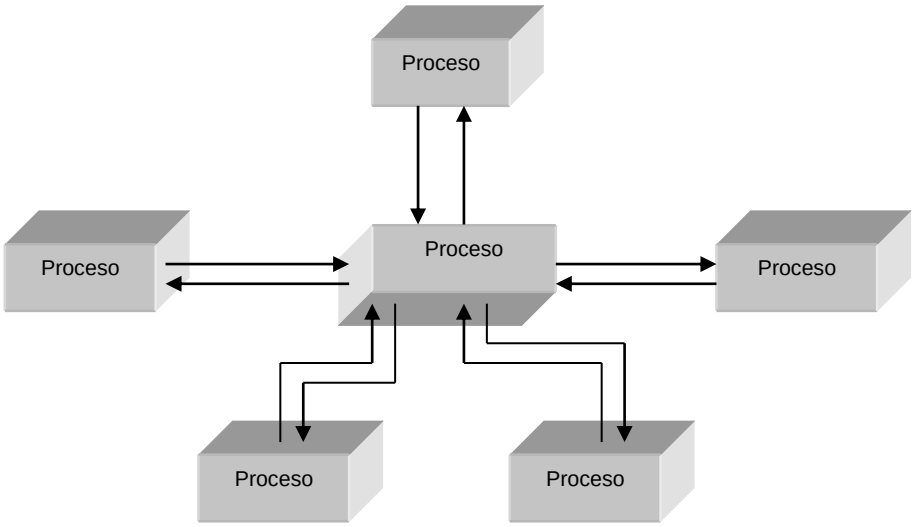

Fig. 21. Farm con un proceso controlador y 5 procesos trabajadores

Con este modelo podría ser interesante observar el rendimiento de la ejecución en paralelo de varios algoritmos de ordenación operando con un mismo conjunto de datos para todos ellos. Si utilizamos como patrón un Farm para resolver el problema planteado, el diseño del modelo sería más o menos el que se muestra en la figura 22.

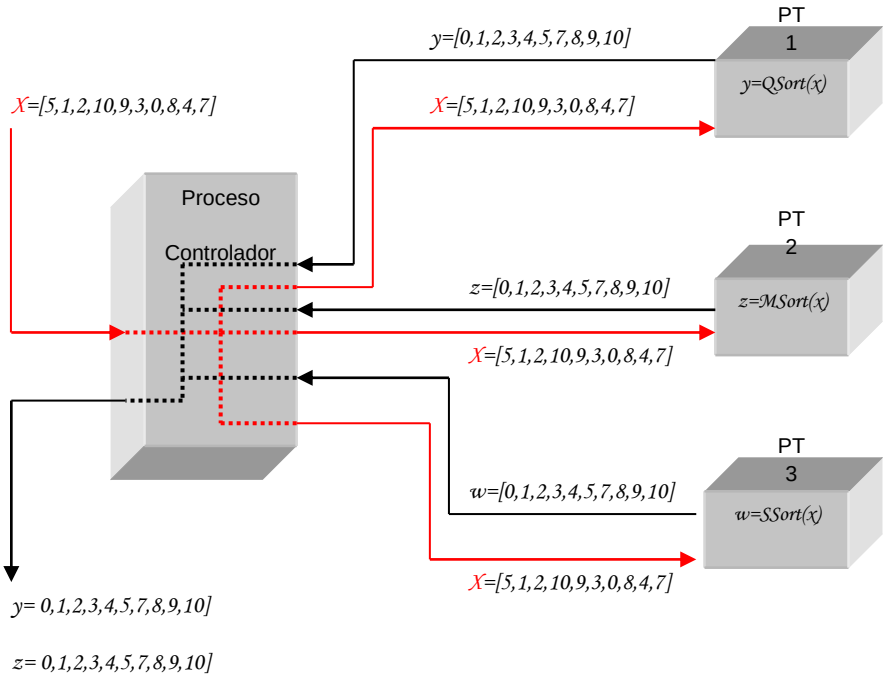

Fig. 22. Farm que ejecuta 3 algoritmos de ordenación en paralelo

El Farm se forma de tres procesos trabajadores (PT1, PT2, PT3) y un proceso controlador. Cada proceso trabajador tiene asociado un algoritmo de ordenación diferente. En el caso del ejemplo de la figura, PT1 tiene asociado el algoritmo QuickSort, a PT2 se le asocia el algoritmo MergeSort y, finalmente, PT3 tiene asociado el algoritmo ShellSort.

El proceso controlador recibe como entrada el conjunto de datos a ordenar, que viene definido como el arreglo X en el ejemplo distribuyendo a sus procesos trabajadores la información, después, en paralelo, se lleva a cabo la ejecución de los tres procesos PT1, PT2 y PT3. El resultado (y, z, w) es devuelto al proceso controlador, que finalmente los retornará como resultados finales también de forma paralela.

Con esta técnica muchos problemas computacionales pueden ser fácilmente paralelizados como un farm de procesos. Algunos ejemplos de estos son:

- el cómputo de matrices;
- el ordenamiento de números;
- la solución de sistemas de ecuaciones;
- la solución al problema de caminos en grafos dirigidos a través de matrices de adyacencia.

El Patrón de Comunicación TreeDV

La técnica de Divide y Vencerás se caracteriza por la división de un problema en subproblemas que tienen la misma forma que el problema completo [BRA98]. La división del problema en subproblemas más pequeños se lleva a cabo utilizando la recursión. El método recursivo continúa dividiendo el problema hasta que las partes divididas ya no puedan dividirse más, entonces se combinan progresivamente y de forma ascendente los resultados parciales de cada subproblema hasta obtener la solución al problema inicial [BRA98]. En esta técnica, la división de cada problema suele hacerse a menudo en dos subproblemas; por tanto, supondremos una formulación recursiva del método Divide y Vencerás con un esquema de división en forma de un árbol binario, cuyos nodos serán procesadores, procesos o threads.

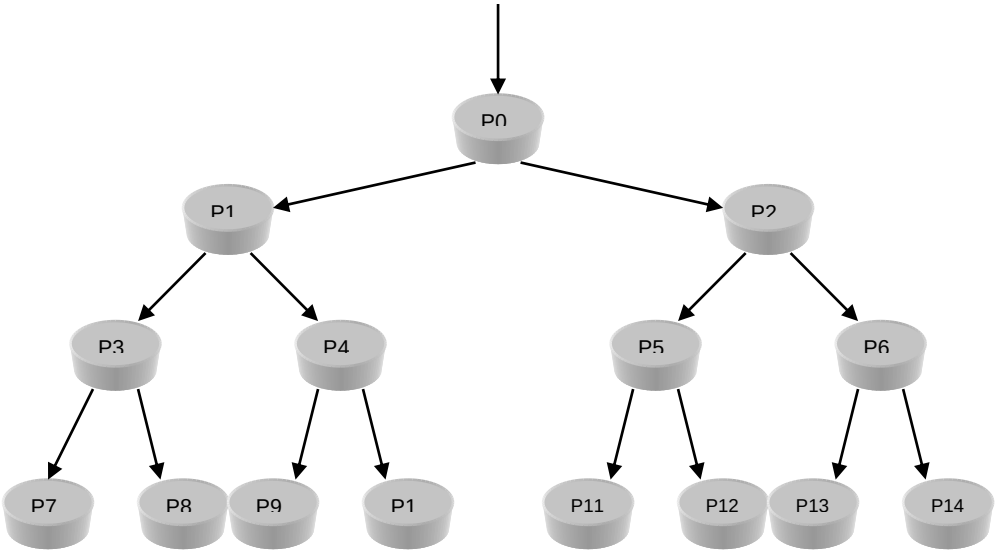

Fig. 23. Árbol binario completo

El nodo raíz del árbol recibe como entrada un problema completo que se divide en dos partes, una se envía al nodo hijo izquierdo, la otra se envía al nodo que representa al hijo derecho (figura. 23). Se repite recursivamente el proceso de división hasta llegar a los niveles más bajos del árbol hasta que, transcurrido un cierto tiempo, todos los nodos hoja reciben como entrada un subproblema de su nodo padre, entonces lo resuelven y le devuelven las soluciones. Cualquier nodo padre en el árbol obtendrá dos soluciones parciales de sus nodos hijos y las combinará para proporcionar una única solución que será la salida del nodo padre. Finalmente el nodo raíz proporcionará como salida la solución completa del problema inicial, [BRI93]. En la figura 23 se muestra un árbol binario completo, esto es, un árbol perfectamente balanceado con los nodos hojas en el mismo nivel, pero podría ocurrir que uno o más nodos hoja aparezcan en diferentes niveles del árbol si el número de subproblemas no es potencia de 2.

Mientras que, en una implementación secuencial, un solo nodo del árbol puede ser ejecutado o visitado cada vez, en una implementación paralela de un algoritmo que siga este esquema se podría visitar más de un nodo al mismo tiempo, es decir, al dividir un problema en dos subproblemas, ambos pueden ser procesados de manera simultánea. Puede ser interesante utilizar esta técnica en una aplicación específica relacionada con el problema de la ordenación de un conjunto de datos, ya que existen varios algoritmos de ordenación que utilizan la técnica de divide y vencerás, junto con la recursión, como en el caso del algoritmo QuickSort u ordenación rápida, o bien en el algoritmo MergeSort, por citar dos de ellos. Supongamos entonces que se tiene una lista de números enteros en desorden y queremos llevar a cabo la ordenación de

dichos números por medio de la técnica de divide y vencerás en paralelo, utilizando para ello el algoritmo de ordenación rápida o Quicksort.

Como primer paso el algoritmo selecciona como pivote uno de los elementos del conjunto de datos que vaya a ordenar; a continuación, el conjunto se divide en dos partes, una a la izquierda y la otra a la derecha del pivote; se desplazan los elementos, de tal manera que los que sean mayores que el pivote queden a su derecha mientras que los que sean menores queden a su izquierda, ver figura 24(a) posteriormente, las partes del conjunto que quedan a ambos lados del pivote se ordenan independientemente de forma paralela y recursiva; se llega al resultado final, que es el conjunto de datos completamente ordenado, ver figura 24(b), si cada nodo del árbol construye el array solución al subproblema que le ha sido encomendado colocando el subarray que le entrega su hijo derecho detrás del que le entrega el izquierdo, antes de devolverla a su nodo padre.

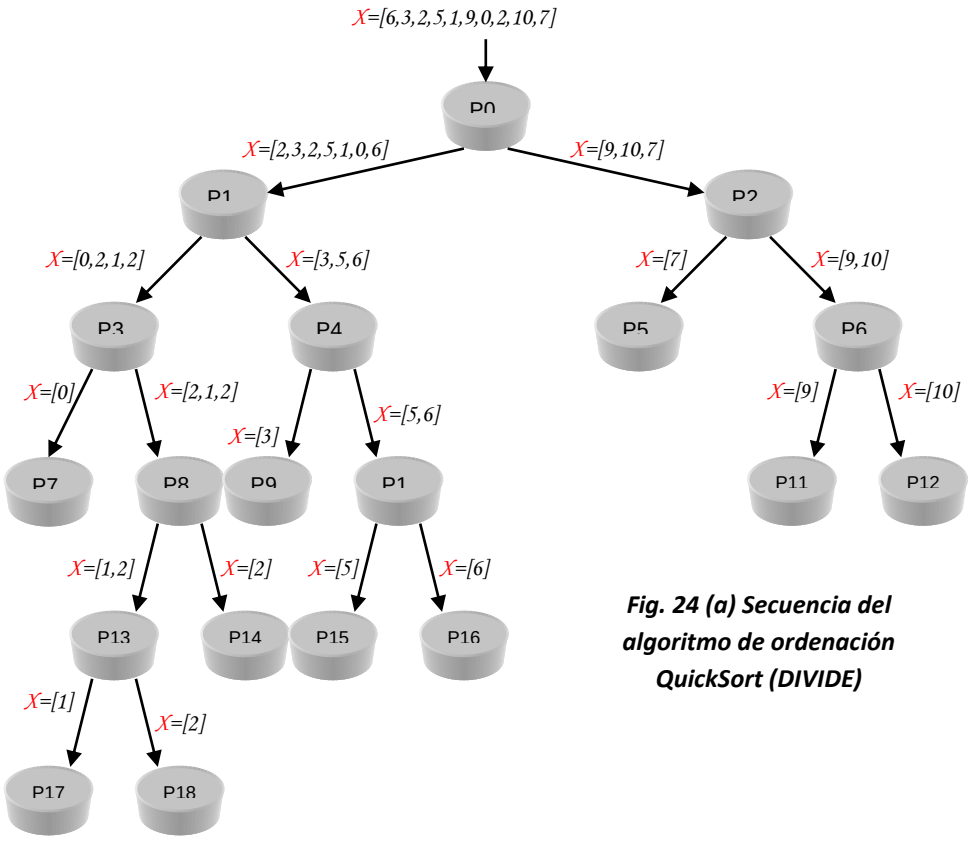

Fig. 24 (a) Secuencia del algoritmo de ordenación QuickSort (DIVIDE)

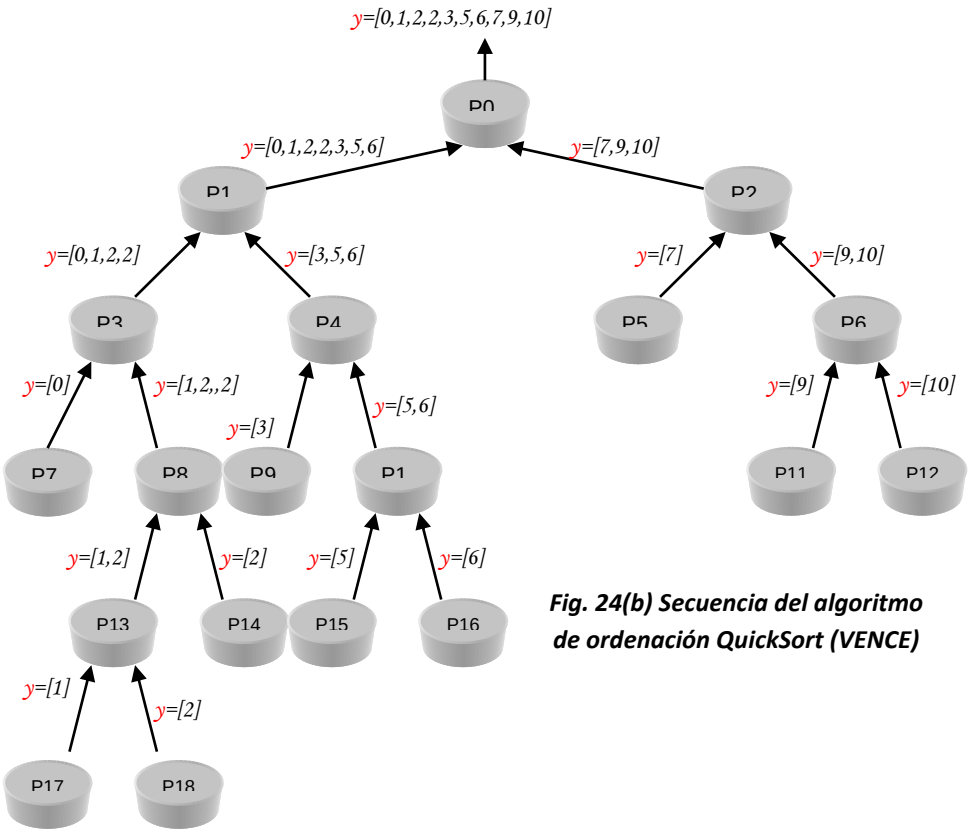

Fig. 24(b) Secuencia del algoritmo de ordenación QuickSort (VENCE)

Con la técnica de divide y vencerás muchos de los problemas computacionales que se llevan a cabo de manera secuencial pueden ser paralelizados como árboles binarios, donde cada nodo representa un proceso y cada proceso del árbol contiene una copia del mismo algoritmo de solución. Ejemplos de problemas que pueden ser solucionados con esta técnica son los siguientes:

- problemas de integración numérica;
- problemas de N-cuerpos;
- problemas de ordenación de datos;
- problemas de búsqueda de datos.