Práctica 6: Algoritmo bucketsort

<table>
<tr><td colspan="2">UNIDAD DE APRENDIZAJE : Aplicaciones para comunicaciones en red
UNIDAD TEMÁTICA IV: Hilos</td></tr>
<tr><td>No. Y Título de la práctica:

Práctica no. 6 Algoritmo bucketsort</td><td rowspan="2">Tiempo de realización: 4.5 horas</td></tr>
<tr><td>Objetivo de la práctica: El estudiante implementará el algoritmo de ordenamiento bucket sort en un entorno distribuido haciendo uso de hilos, así como sockets de flujo.</td></tr>
<tr><td colspan="2">Situación problemática: En muchos tipos de aplicaciones se requiere aplicar técnicas de ordenamiento de datos, por ejemplo, en procesamiento de imágenes existen muchos tipos de filtros que hacen uso de estas técnicas, sin embargo, cuando la cantidad de datos a ordenar se vuelve demasiado grande, algoritmos como el quicksort se vuelven poco eficientes, simplemente porque no es posible almacenar tal cantidad de datos en un arreglo. En estos casos el algoritmo bucketsort es una buena alternativa, ya que puede ser paralelizado y la carga de procesamiento puede ser distribuida en varios servidores para acelerar el ordenamiento. La única condición es que la distribución de los números sea uniforme.

¿Para ordenar grandes cantidades de números qué método de ordenamiento será más adecuado utilizar en conjunto con bucketsort? ¿Por qué?</td></tr>
<tr><td colspan="2">Competencia específica: Desarrolla aplicaciones en red, con base en el modelo cliente-servidor y utilizando de sockets de flujo, así como hilos para el envío de datos a cada uno de los servidores.</td></tr>
<tr><td>Competencias genéricas:
Aplica los conocimientos en la práctica
Demuestra habilidad para trabajar en equipo
Demuestra capacidad de investigación
Desarrolla aplicaciones en red con base en la tecnología más adecuada</td><td>Elementos de competencia:
Programa aplicaciones en red con base en el modelo Cliente-Servidor y la interfaz de aplicaciones de sockets de flujo
Analiza los servicios definidos en la capa de transporte
Emplea el modelo Cliente-Servidor para construir aplicaciones en red
Programa aplicaciones Cliente-Servidor utilizando sockets de flujo
Programa aplicaciones utilizando hilos de ejecución para distribuir tareas a distintos servidores</td></tr>
</table>

Introducción

El ordenamiento por cubetas (bucket sort en inglés) es un algoritmo de ordenamiento que distribuye todos los elementos a ordenar entre un número finito de cubetas. Cada cubeta sólo puede contener los elementos que cumplan unas determinadas condiciones. Para esta práctica esas condiciones son intervalos de números. Las condiciones deben ser excluyentes entre sí, para evitar que un elemento pueda ser clasificado en dos cubetas distintas. Después cada una de esas cubetas se ordena individualmente con otro algoritmo de ordenación (que podría ser distinto según la cubeta), o se aplica recursivamente este algoritmo para obtener cubetas con menos elementos. Se trata de una generalización del algoritmo Pigeonhole sort. Cuando los elementos a ordenar están uniformemente distribuidos la complejidad computacional de este algoritmo es de O(n).

El algoritmo contiene los siguientes pasos:

1. Crear una colección de cubetas vacías
2. Colocar cada elemento a ordenar en una única cubeta
3. Ordenar individualmente cada cubeta
4. Devolver los elementos de cada cubeta concatenados por orden

Recursos y/o materiales

- Manual de prácticas de laboratorio de Aplicaciones para Comunicaciones en Red
- Plumones
- Bibliografía
- Internet
- Computadora
- IDE de desarrollo
- Apuntes

Instrucciones

En esta práctica debes implementar el algoritmo de ordenamiento bucketsort en un modelo cliente-servidor, haciendo uso de un servidor para el ordenamiento de cada cubeta.

Desarrollo de la práctica

A partir de los programas hilo_retorno, LinServer y LinClient que te serán proporcionados por el profesor deberás realizar las siguientes modificaciones:

- El programa hilo_retorno contiene el código para generar un hilo, este invocará una función, a la cual se le pasa una estructura de datos como parámetro de entrada y devolverá un dato como salida de la función. Deberás realizar las siguientes modificaciones:

  - Se generará una lista (o arreglo) con 3500 números 'N' enteros de forma aleatoria en el rango de 0-999, los cuales deberán ser ordenados usando el algoritmo bucketsort. Para esto, el usuario deberá elegir el número de cubetas 'C' a ser utilizadas (1…3500).
  - Se crearán 'C' cubetas para insertar en ellas los números de la lista inicial siempre y cuando ajusten con el rango de la cubeta(Ejemplo. Si se usan 5 cubetas y los números están en el rango del 0-999, entonces cada cubeta tendrá un rango uniforme de 200 números: 0-199 para la cubeta1, 200-399 para la cubeta 2, etc.).
  - Una vez creadas las cubetas, estas deberán ser enviadas para su ordenamiento en un igual número 'C' de servidores de manera concurrente a través del uso de hilos. Modifica el cuerpo de la función invocada por cada hilo, para que dentro de ésta se cree un cliente y envíe una de las cubetas a ser ordenadas al servidor correspondiente.

  Sugerencia. Modifica la estructura que se pasa al hilo, para que contenga la siguiente información:

  1. Dirección del servidor que recibirá la cubeta
  2. Puerto de la aplicación servidor que recibirá la cubeta
  3. Cubeta de n elementos a ser ordenada
  4. Tamaño de la cubeta

  Antes de crear los hilos, primero deberás

- El programa LinServer implementa un servidor de flujo con hilos de ejecución. Deberás modificar el código para que reciba la cubeta con los datos a ser ordenados y una vez ordenados, estos deberán ser devueltos al cliente. Puedes implementar el método de ordenamiento que gustes

- El programa LinClient contiene un cliente de flujo común y corriente. Puedes usarlo como base para hacer las modificaciones que se te piden en el programa hilo_retorno.