Informe sobre la implementación de algoritmo de ordenamiento distribuido ‘QuickSort’ con llamadas asincrónicas.

Para iniciar, se abordará el concepto general del algoritmo. QuickSort, se basa en la estrategia divide y vencerás, el cual consiste en descomponer un problema en serie de subproblemas de menor tamaño, resolverlos recursivamente y combinar las soluciones para obtener una general (que es la solución del problema original). Así, el algoritmo, al dividir, realiza comparaciones para hacer correcta la división de subproblemas: elementos menores o iguales al objeto a comparar (pivote), a la derecha, y elementos mayores al pivote a la izquierda.

Este proceso estará guiado entre un Servidor y un Cliente. Así, el cliente inicialmente le otorgará la lista de elementos a ordenar al Servidor. Y éste último se encargará de realizar los procesos correspondientes de QuickSort sobre la lista con ayuda de sub-servidores; cada sub-server tendrá una parte de la lista otorgada por el Server, así cada uno se comprometerá en resolver/ordenar la sub-lista otorgada y entregársela correctamente a su Servidor. Finalmente, el Servidor tendrá la lista completa nuevamente, gracias a sus servidores ayudantes, y podrá darle entrega de la solución ordenada al Cliente.

El algoritmo QuickSort tiene una complejidad algorítmica promedio de , donde es el número de elementos en la lista. Esto se debe a que divide la lista en dos partes y realiza la operación de ordenamiento en cada una de ellas de manera recursiva. Sin embargo, se vuelve de menor complejidad al ser distribuido al ser afectado por una cantidad de servidores, la complejidad quedaría determinada como: .

Para probarlo, se utilizó una lista de 1000 números sin orden y de manera randómica, en el que se pudo comprobar la correctitud de la ejecución del programa para ordenarlos, para ello se demoró 70 milisegundos en realizarlo. De todas maneras, para probar el funcionamiento del mismo, se utilizaron diferentes tamaños de listas con diferentes tipos de números. Solo que para una prueba final, se puede probar el caso más prometedor (que es el ya mencionado).

Conclusión.

La implementación del algoritmo QuickSort en un entorno distribuido con llamadas asíncronas ha demostrado ser eficiente y efectiva. La estrategia de dividir y vencer, junto con la distribución de tareas entre varios servidores, permite un procesamiento más rápido y eficiente de los datos. Donde su complejidad algorítmica promedio se mantiene, la distribución de la carga de trabajo entre varios servidores puede reducir efectivamente el tiempo de ejecución. Sin embargo, es importante tener en cuenta que la sobrecarga de comunicación entre el servidor y los sub-servidores puede afectar el rendimiento general.