Name : Kiran K. Patil   Course : Parallel computing Lab

ID : 211070904   Sem : 06

Assignment 03

Aim : Implementation of parallel quick sort [Hyper Quice sort] using CUDA.

Theory : The traditional way to implement quicksort on serialized processors is where--by steps are (sorted) executed sequentially untill the program terminates. Quicksort

Quicksort is a highly efficient algorith /sorting technique that devides a large data array in smaller ones. A vast array is divided into two arrays, one containing values smaller than the provided value, say pivote, on which the partition is based and the other contains values greater than pivote value.

While executing quicksoot on cpu one proc gets started and it executes the code line by line.

parallel programming is whereby a program is broken down into concerment programs which are executed concurrenty on multiple threads on a processor. Here the coordination is required.

There are many approaches to implement quicksort algorithm parallely.

1) Naive parallel quick sort.
2) Optimized parallel quick sort.
3) Using both sequential and parallel approaches.
4) Hyperquick sort
5) parallel quicksort by regular sampleting

- Hyperquicksort :- This approach is an improvment of both sequential and parallel approache There was a problem of load balancing This process improves the chances of finding a true median by sorting the subjects sequentially using one pivote that is broadcasted to all the processes at the beginning of the algorithm.

- Steps :-

1) A list of size n is divided among 'n' processes. Assume List of size 16 & 4 proce so each process will handle 4 elements.

2) A process among the four responsible for finding the pivote element, finds a pivote and broadcast it to all the processes which sort their the (algorithm) sublists sequentially using the broadcasted pivote element. This step will improve chances of finding pivotes close to the true medium.

3) Pivote selection and broadcasting to another processes. Sublist partitioning of low and high values. swapping of values between partner- processes.

4) The remaing top half from one partner processes (sublist partitioning of low and high) and the received top half from the other partner process are merged. into local sublist for each process.

5) Recurse the upper half and lower half of each subprocess to archive a sorted list.

6) finally merge the processes in order to get fully sorted list

Analysis : There are log(n) steps and n processe the total time complexity is O(log n). Space complexity is O(log n)

Conclusion:- Thus, we have successfully implemented quicksort

2) A process among the four responsible for finding the pivote element., finds a pivote and broadcast it to all the processes which sort their the (algorithm) sublists sequentially using the broadcasted pivote element. This step will improve chances of finding pivotes close to the true medium.

3) Pivote selection and broadcasting to another processes. Sublist partitioning of low and high values. swapping of values between partner-processes.

4) The remaing top half from one partner processes (sublist partitioning of low and high) and the received top half from the other partner process are merged. into local sublist for each proces.

5) Recurse the upper half and lower half of each subprocess to archive a sorted list

6) finally merge the processes in order to get fully sorted list

Analysis: There are log(n) steps and n processe the total time complexity is O(log n). space complexity is O(log n)

Conclusion:- Thus, we have successfully implemented quicksort