Assignment 3

Nachiket Erlekar,41434

August 29, 2020

1 Title

Sorting techniques using parallel computing

2 Objective

- Learn parallel decomposition of problem.
- Learn sorting algorithms.

3 Problem Statement

Sorting Operations-
Design parallel algorithm for sorting techniques like:

- Bubble sort
- Merge sort

4 Outcome

We will be able to decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.

5 Software and Hardware requirements

1. Operating System : 64-bit Linux or its derivative
2. Programming Language: C/C++
3. Google colab
4. Openmp

6 Date of Completion

August 29, 2020

7 Assessment grade/marks and assessor's sign

8 Theory- Concept in brief

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called decomposition. Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem. Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size.

n blocks and one thread per block.

1 block and n threads in that block.

m blocks and n threads per block.

The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. The OpenMP API defines a portable, scalable model with a simple and flexible interface for developing parallel applications on platforms from the desktop to the supercomputer.

8.1 Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list.

procedure bubbleSort(A : list of sortable items)
n := length(A)
repeat
swapped := false
for i := 1 to n - 1 inclusive do
if A[i - 1] ¿ A[i] then

swap(A[i - 1], A[i])
swapped = true
end if
end for
n := n - 1
until not swapped
end procedure

8.2 Merge Sort

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

MergeSort(arr[], l, r):
If r is less than l
1. Find the middle point to divide the array into two halves:
middle m = (l+r)/2
2. Call mergeSort for first half:
Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
Call merge(arr, l, m, r)

9 Test cases

Bubble sort program:
83 86 77 15 93 35 86 92 49 21
Result(parallel) :
15 21 35 49 77 83 86 86 92 93
Time parallel = 0.000136226

Merge sort program:
83 86 77 15 93 35 86 92 49 21
15 21 35 49 77 83 86 86 92 93
Time parallel = 0.00245084

10 Conclusion

We have successfully conducted bubble sort and merge sort based on existing sequential algorithms, design and implemented parallel algorithm utilizing all resources available.