PARALLEL SUM REDUCTION

The goal of this work is to obtain the sum of all the values of a large 1D array using Parallel Sum Reduction Operator. There are number of ways for computing the sum of the elements of an array but parallel sum reduction approach seems to be a better solution than sequential approach. Here basically we divide the array into multiple partitions, each thread will calculate the sum of the elements reside in that partition and finally adds up the results. Reduction operator reduce the collection of values into a single value.

PARALLEL SUM REDUCTION USING OpenCL

OpenCL: stands for Open Computing Language. It is a framework for general purpose computation on different systems. It is most commonly used on General purpose computing but can also be implemented on multiple core CPUs.

Implementation: Parallel Sum Reduction computation is done using OpenCL. Below picture explains the algorithm behind parallel sum computation implemented in this work.

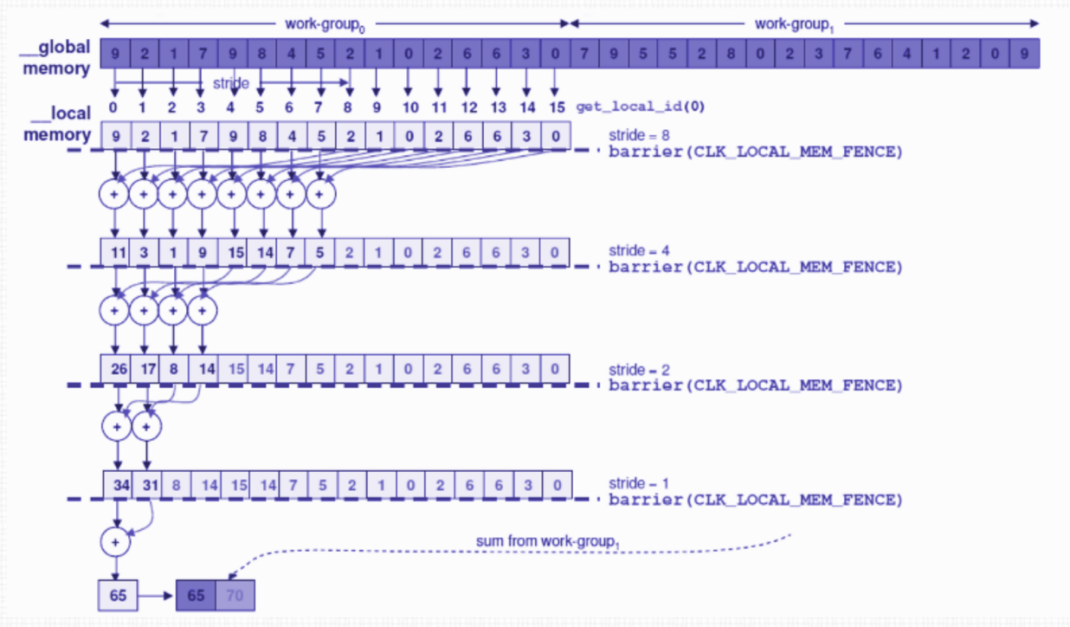

Image 2: Parallel Sum Reduction logic

Here we basically keep array size equal to the number of work items. Number of work items that is group size should be evenly divisible. We know that each work group will have its own memory and it is local to the work items in that particular group. Each work group is divided into two equal parts and first portion of the work group is added to other. This process is repeated until we left with only one element which is sum of all the values in each work group. We need to make sure that all the threads within same workgroup are synchronized, this is achieved using an instruction barrier (CLK_LOCAL_MEM_FENCE). Now we have computed sum values of each work group. These values will be added sequentially to get sum of all the values inside an array. Source code for the same is provided with files named *main.c* and *sumReductionGPU.cl*

Results: The following table shows the **execution time in micro seconds** for various input array size.

Input Array Size	**Work group size**	**Time taken for sequential in micro seconds**	**Time taken for parallel (OpenCL) in micro seconds**
262144	256	556	462
524288	256	2236	1553
1048576	256	2195	1506
2097152	256	4817	2438
4194304	256	10850	6465
16777216	256	41367	17043
134217728	256	318918	110466

Table 1: Results of OpenCL for various input size