High Performance Computing Report
Lab 5

Amarnath Karthi (201501005)
Chahak Mehta (201501422)

DA-IICT

1 Image warping

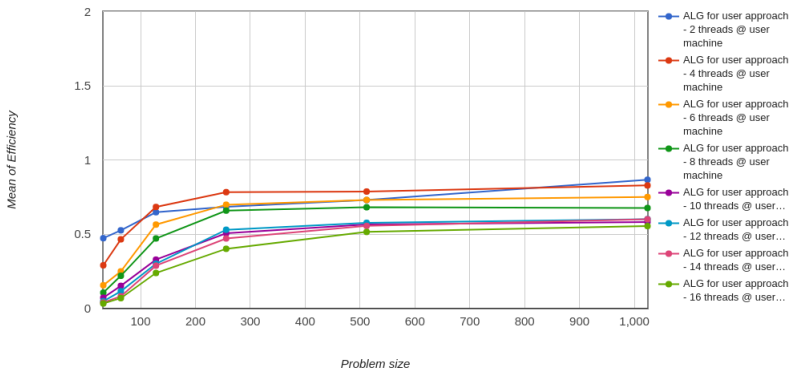

Fig. 1. Efficiency vs problem size

Implementation Details

1(a) Brief and clear description about the Serial implementation

The test image is warped by using the principle of rotation and changing the angle of rotation (theta) according to the number of pixels/distance.

1(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

The serial code is then parallelized by using OpenMP parallel directive and by manually assigning the threads the chunks of the image the thread has to work on thereby working on the principle of data division

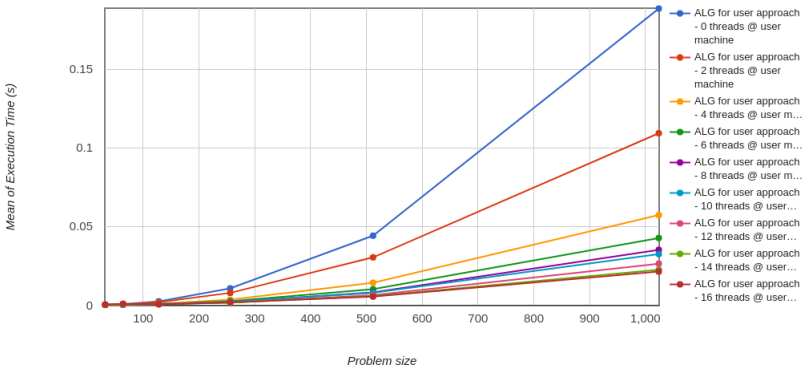

Fig. 2. Execution time vs problem size

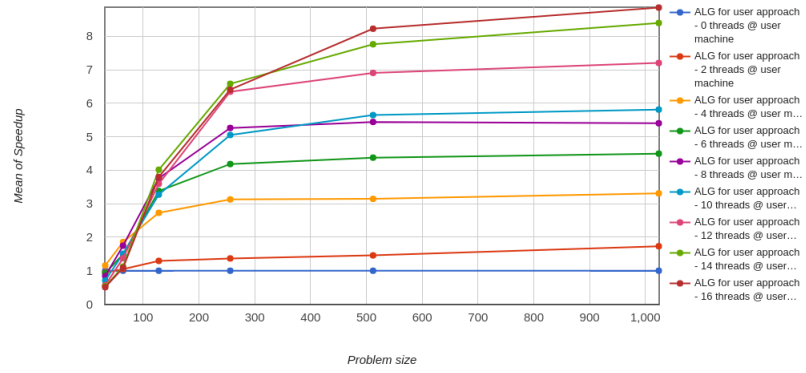

Fig. 3. Speedup vs problem size

1(c) Time Curve related analysis (as no. of processor increases)

We can see from Figure 1 that the time taken by the parallel program to run decreases as the number of cores that perform the operations. This is due to the division of the work to be done among more number of available cores which in turn leads to more parallel operations to be performed, thereby decreasing the execution time for the code.

1(d) Time Curve related analysis (as problem size increases, also for serial)

As the problem size increases, the time taken by the serial code increases exponentially. The time taken by the parallel code increases at a much slower rate because of the reasons mentioned in the above answer.

1(e) Speedup Curve related analysis (as problem size and no. of processors increase)

We can see in the Speedup curve that the parallel code is faster by a factor of maximum 9, unlike the theoretical factor of 16. This is due to the increase in the parallel overhead with the increase in the number of threads/processors performing the operations

1(f) Efficiency Curve related analysis

The efficiency for cores can be calculated by $\frac{speedup}{numberofprocessors}$ This doesn't reach 1 because of the fact that the speedup is much lower than the theoretical value of the speedup for a particular number of cores

2 Image median filtering

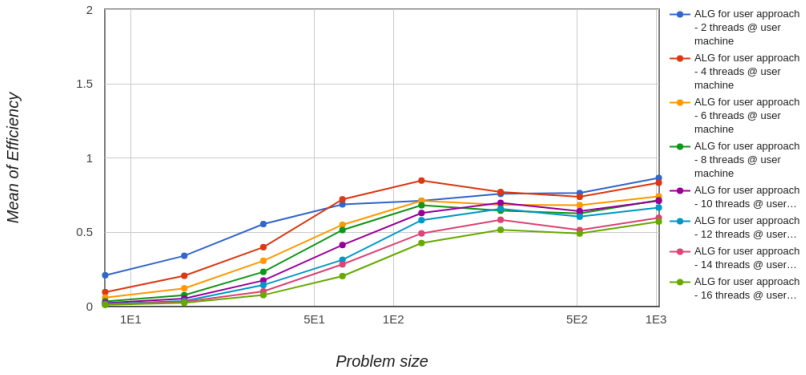

Fig. 4. Efficiency vs problem size

Implementation Details

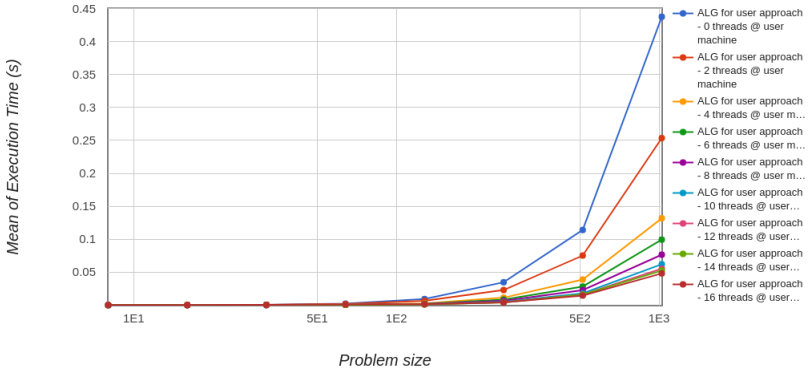

Fig. 5. Execution time vs problem size

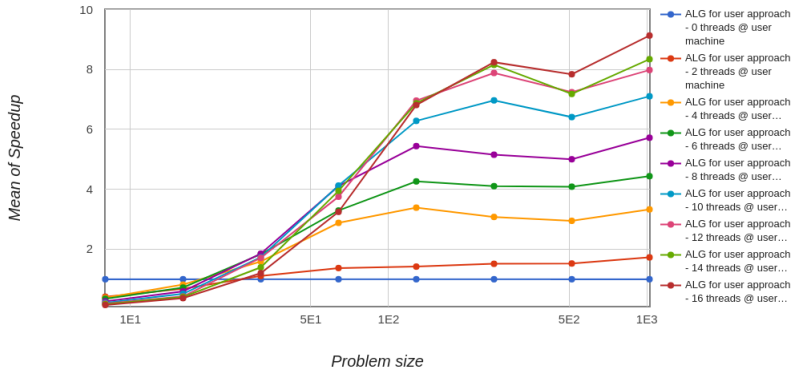

Fig. 6. Speedup vs problem size

2(a) Brief and clear description about the Serial implementation

The test image is filtered using the concept of median filtering. For this, the data values of the pixels are determined by the median of the values of the pixels in the neighbourhood of a specified halfwidth. Increasing the size of the halfwidth will increas the time taken to run the code since it increases the number of pixels to be taken into account to determine the median of a neighbourhood of a pixel which will slow down the code considerably.(This technique can also be used to filter out noise from noisy images or as an enhancement filter on other images).

2(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

The serial code is parallelized by using OpenMP parallel directive and by manually assigning the threads the chunks of the image the thread has to work on thereby working on the principle of data division

2(c) Time Curve related analysis (as no. of processor increases)

We can see from Figure 4 that the time taken by the parallel program to run decreases as the number of cores that perform the operations. This is due to the division of the work to be done among more number of available cores which in turn leads to more parallel operations to be performed, thereby decreasing the execution time for the code.

2(d) Time Curve related analysis (as problem size increases, also for serial)

As the problem size increases, the time taken by the serial code increases exponentially. The time taken by the parallel code increases at a much slower rate because of the reasons mentioned in the above answer.

2(e) Speedup Curve related analysis (as problem size and no. of processors increase)

We can see in the Speedup curve that the parallel code is faster by a factor of maximum 9, unlike the theoretical factor of 16. This is due to the increase in the parallel overhead with the increase in the number of threads/processors performing the operations

2(f) Efficiency Curve related analysis

The efficiency for cores can be calculated by $\frac{speedup}{numberofprocessors}$ This doesn't reach 1 because of the fact that the speedup is much lower than the theoretical value of the speedup for a particular number of cores

3 Image normalization

Implementation Details

3(a) Brief and clear description about the Serial implementation

The test image is normalized by using the formula to standardize a Gaussian variable. This is done by calculating the Mean and Standard deviation of the values of the pixels of the image and then by the formula $Z = \frac{X-\mu}{\sigma}$.

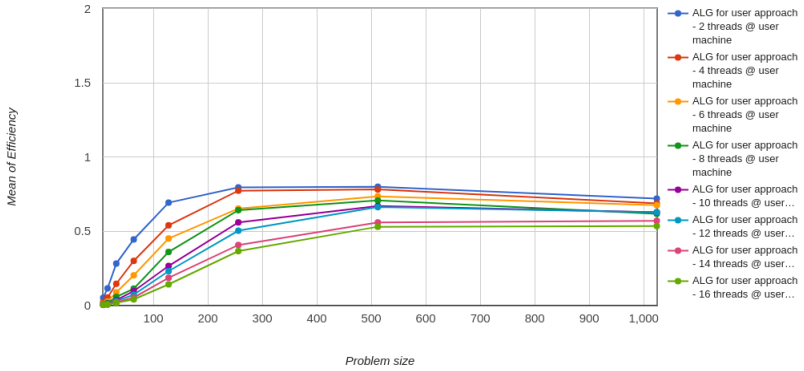

Fig. 7. Efficiency vs problem size

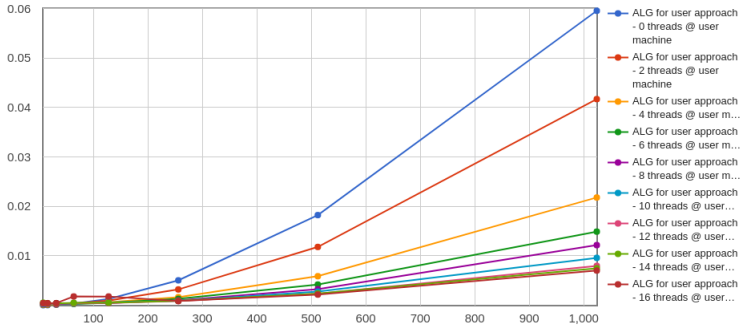

Fig. 8. Execution time vs problem size

3(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

The serial code is then parallelized by using OpenMP parallel directive and by manually assigning the threads the chunks of the image the thread has to work on thereby working on the principle of data division

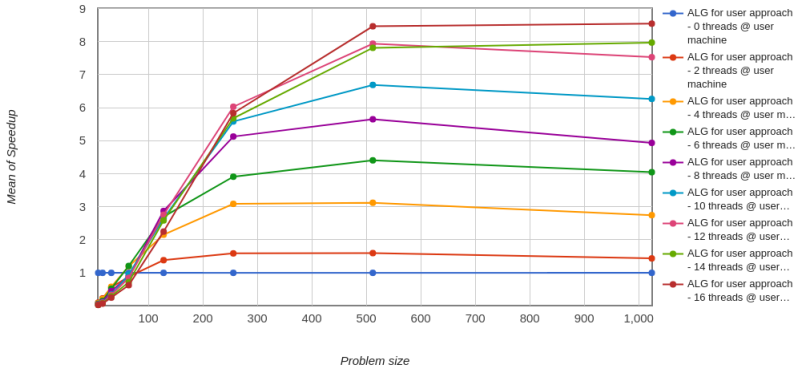

Fig. 9. Speedup vs problem size

3(c) Time Curve related analysis (as no. of processor increases)

We can see from Figure 7 that the time taken by the parallel program to run decreases as the number of cores that perform the operations. This is due to the division of the work to be done among more number of available cores which in turn leads to more parallel operations to be performed, thereby decreasing the execution time for the code.

3(d) Time Curve related analysis (as problem size increases, also for serial)

As the problem size increases, the time taken by the serial code increases exponentially. The time taken by the parallel code increases at a much slower rate because of the reasons mentioned in the above answer.

3(e) Speedup Curve related analysis (as problem size and no. of processors increase)

We can see in the Speedup curve that the parallel code is faster by a factor of maximum 9, unlike the theoretical factor of 16. This is due to the increase in the parallel overhead with the increase in the number of threads/processors performing the operations

3(f) Efficiency Curve related analysis

The efficiency for cores can be calculated by $\frac{speedup}{numberofprocessors}$ This doesn't reach 1 because of the fact that the speedup is much lower than the theoretical value of the speedup for a particular number of cores

4 Additional Image Results

Fig. 10. Median filtering on an image of resolution 1024x1024

Fig. 11. Normalisation on an image of resolution 1024x1024