Introduction to Parallel & Distributed Programming COL380 -Programming Assignment 1

Ankesh Gupta
2015CS10435

Parallel Convex Hull

Data Structures and Design Decisions

1. The algorithm was adapted from *Quickhull Algorithm* algorithm on *Wikipedia*.
2. Its uses a *divide and conquer* to locate mandatory hull points, and then discard points lying on the inside of enclosing rectangle.
3. The program has been attempted to parallize by making the recursive *quickhull* call being run on an independent thread.
4. Also, call to recursive function inside *pragma* construct was made *non blocking/no wait*, so as to allow other threads to work in parallel.
5. Data Structures used were vectors, a *dynamic-array*. The reason was because it is mutable and doesnt require size declaration as beginning.
6. To limit the *number of threads*, a global counter was maintained, shared by all threads, which tracked how many threads are available.
7. Each time a thread get created/destroyed, an access is made to the variable(mentioned above), which according to availability, may either accept/reject request.
8. If threads are not available, the remaining program becomes sequential.

Analyzing Graphs

1. The efficiency curve follows a similar trend as it was in prefix sum case: Decreases with increase in thread(for any problem size).
2. Speedup curve is different. Never is the speedup of parallel implementation > 1.
3. This reflects that the algorithm is indeed suffering from large serial component, which can't be parallelised.
4. One of the major time consuming operation is generation of the *candidate hull points* from the image.
5. Passing vector references would add on to speedup of the program.
6. But, one trend persists. With increase in problem size, speedup acheived showed an increasing trend(though less than 1). This is consistent with ***Gustafson's Law***.

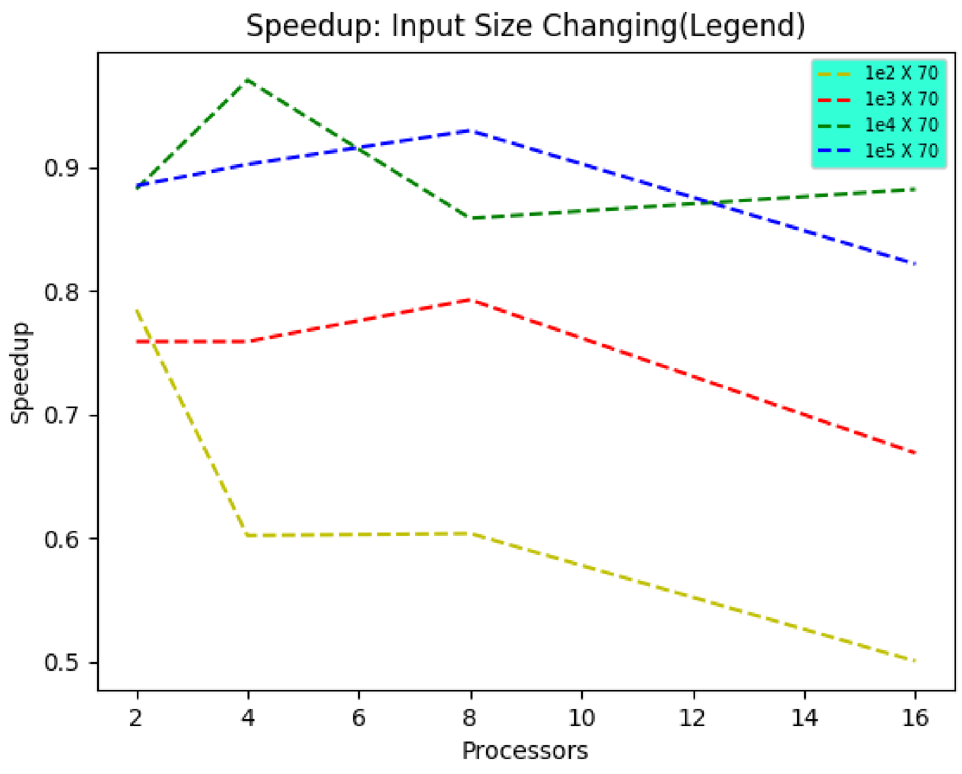

Figure 1: Speedup vs Processors for Varying input

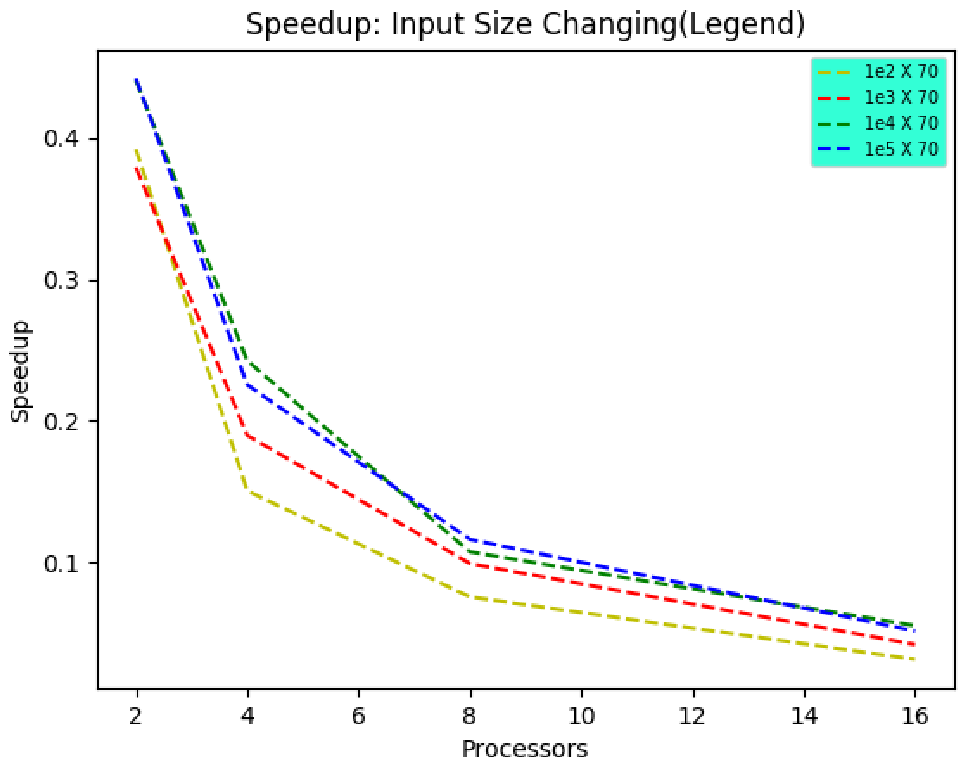

Figure 2: Efficiency vs Processors for Varying input