Submitted by:

Md. Tahrim Hossain

Roll - 48

The formula for speedup (S) is commonly given by:

$S = T_s/T_p$

where:

- T_s is the execution time of the task when performed sequentially (using a single processor or core).
- T_p is the execution time of the task when performed in parallel using multiple processors or cores.

A speedup value greater than 1 indicates an improvement in performance

Superlinear speedup refers to a scenario in parallel computing where the performance of a parallel algorithm or system exceeds the theoretically expected speedup. In other words, the speedup achieved is greater than the ratio of the number of processors used.

Sublinear speedup occurs when the performance improvement gained by using parallel processing is less than what would be expected based on the number of processors employed. In other words, the speedup achieved is lower than the linear speedup predicted by Amdahl's Law or other theoretical models.

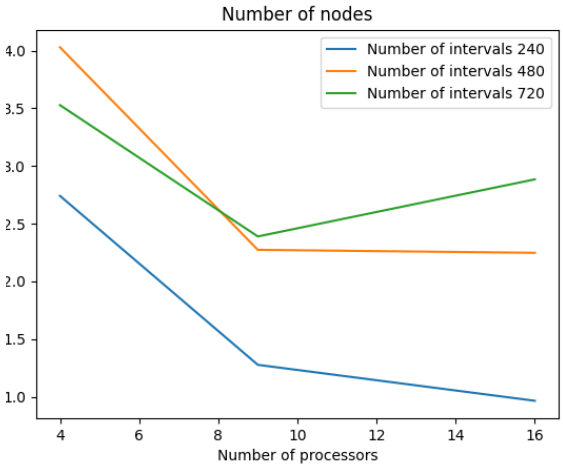

Fig: speedup of parallel version

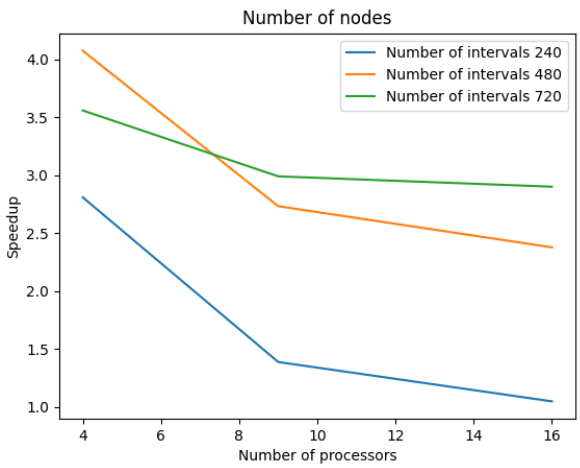

Fig: speedup of pipeline version

Here we can see both cases of superlinear and sublinear speedup. Several reasons could contribute to superlinear speedup:

1.Caching Effects: With a larger number of processors, there may be increased opportunities for data caching. This can lead to reduced memory access times, resulting in faster overall execution.

2.Resource Availability: Additional processors may bring extra resources, such as more cache or memory bandwidth, into play. This abundance of resources can enhance overall system efficiency and contribute to superlinear speedup.

3.Load Balancing: Efficient load balancing can lead to superlinear speedup. If tasks are dynamically allocated to processors in a way that optimally utilizes their capabilities, the overall execution time can be reduced more than expected.

4.Communication Overhead Reduction:If the communication overhead in a parallel system is reduced with more processors, it can result in a better-than-linear speedup. This may happen when the communication cost per processor decreases as the number of processors increases.

6.Parallel I/O Improvements:If the parallel algorithm involves input/output operations, a larger number of processors might lead to better parallelization of I/O, resulting in improved overall performance.

We can also see sublinear speedup in these graphs. Several reasons could contribute to sublinear speedup:

1.Communication Overhead:As the number of processors increases, the communication between them can become a significant bottleneck. The time spent on coordinating and exchanging information among processors may grow faster than the benefits gained from parallelization, leading to sublinear speedup.

2.Synchronization Delays:Parallel algorithms often involve synchronization points where processors need to wait for others to complete certain tasks. If synchronization becomes a limiting factor and introduces delays, it can impede the overall speedup.

3.Contention for Resources:As more processors are added, contention for shared resources such as memory or I/O can increase. This contention can lead to contention delays, reducing the efficiency of parallel execution and causing sublinear speedup.

4.Amdahl's Law Limitations:Even if a portion of a program is parallelizable, there may be sequential components that cannot be parallelized. Amdahl's Law dictates that the speedup of a

program is limited by the fraction of the code that cannot be parallelized. If this non-parallelizable fraction is significant, it can result in sublinear speedup.

5.Algorithmic Constraints:The nature of the algorithm itself may impose limitations on parallelization. Some algorithms inherently have dependencies or sequential components that prevent efficient parallel execution.

6.Scalability Issues:Not all algorithms or problems scale well with the number of processors. Some problems might exhibit diminishing returns as the number of processors increases, leading to sublinear speedup.

7.Resource Contention:In addition to contention for shared resources, there can be contention for computational resources. If multiple processors are competing for processing units, the overall performance gain may not scale linearly with the number of processors.