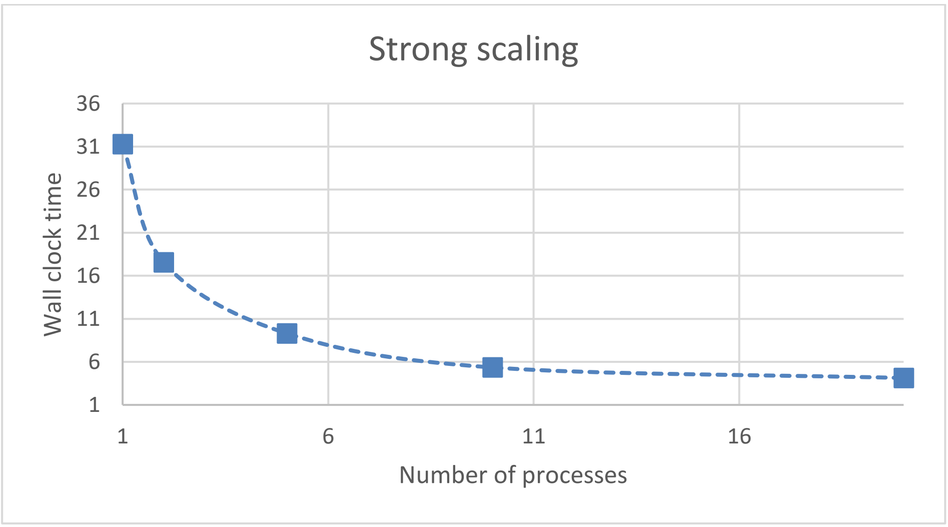
The Following is the report of weak and strong scaling tests done on a MPI parallelized Conjugate Gradient solver. Along with comparison of this solver with Gauss-seidel and Multi Grid algorithms.

*Table 1 Strong scaling*

| Number of MPI Processes | Wall-Clock Time (s) |
|---|---|
| 1 | 31.304707237 |
| 2 | 17.55697811 |
| 5 | 9.270324119 |
| 10 | 5.34354532 |
| 20 | 4.12561892 |

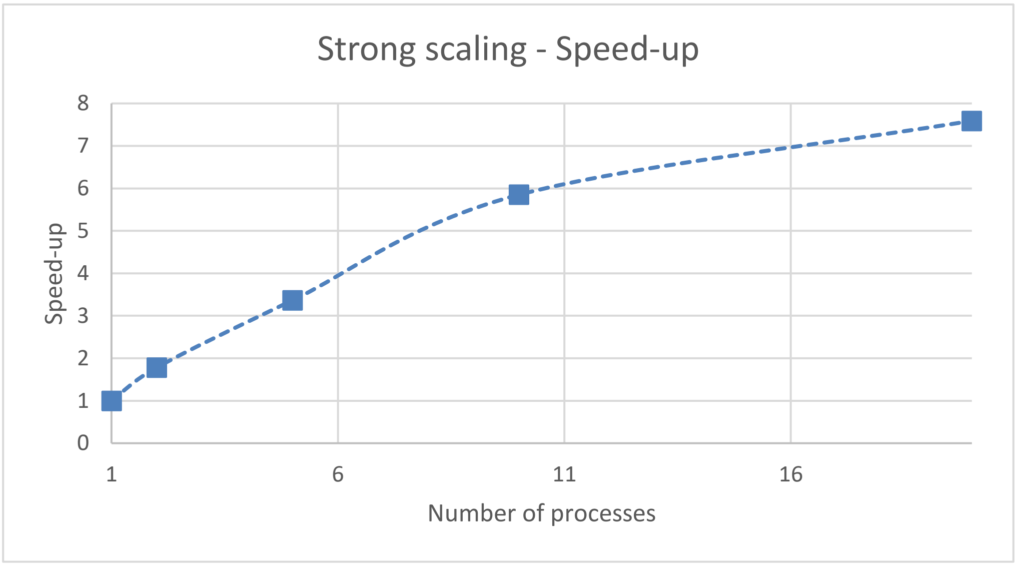
Table 1.0 shows how the wall-clock time of 4 iterations of the conjugate gradient method decreases as the number of MPI processes increases for a fixed problem size (18000x18000 grid). It demonstrates the benefits of parallelization, as the execution time significantly reduces with more processes.

As Graph 1 illustrates, as number of processes increases, the wall-clock time decreases significantly, indicating a good speed-up. It shows a decreasing curve, illustrating how the wall-clock time drops as more processes are added. The graph represents the speed-up achieved through parallelization. However, with higher number of nodes, we see the drop in run time to decrease in value, which could indicate we have reached the optimum number of processes for this problem domain size.

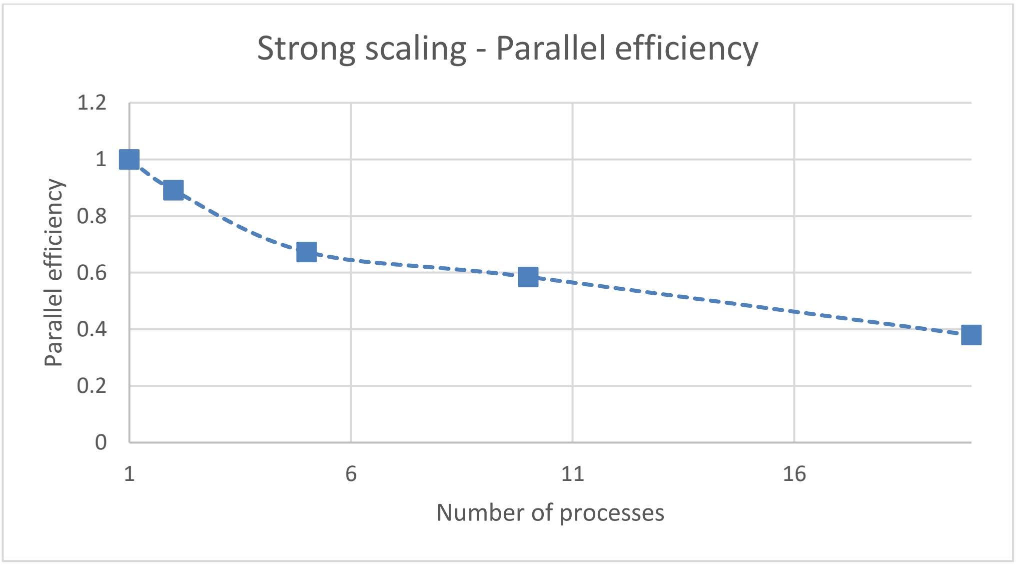


Graph 1.0 Wall-Clock Time vs Number of Processes

Graph 2.0 displays the speed-up factor achieved by increasing the number of processes. It helps quantify how much faster the program runs with more parallel resources, in comparison with a serial run. The parallel efficiency, illustrated in graph 3.0, decreases with an increasing number of processes, it shows how the efficiency of parallelization changes as more processes are added. Efficiency decreases with more processes, which is common due to communication overhead and load balancing issues.



Graph 2.0 Speed Up

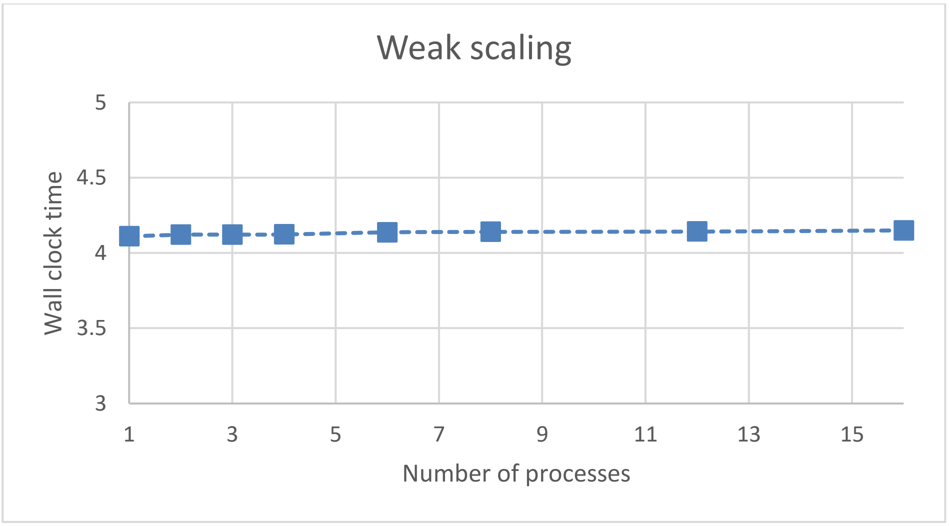


Graph 3.0 Parallel Efficiency

*Table 2 Weak Scaling*

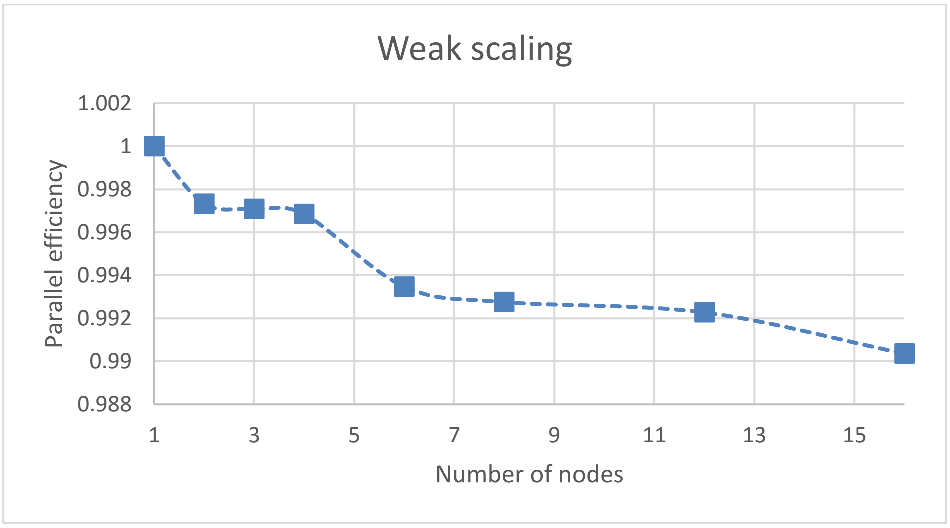
| #nodes (Np=nodes*20) | I (Nx=I*18000) | J (Ny= J*18000) | Runtime |
|---|---|---|---|
| 1 | 1 | 1 | 4.119294903 |
| 2 | 2 | 1 | 4.121384359 |
| 3 | 3 | 1 | 4.122032329 |
| 4 | 2 | 2 | 4.123066491 |
| 6 | 2 | 3 | 4.137786122 |
| 8 | 2 | 4 | 4.140833586 |
| 12 | 3 | 4 | 4.142065309 |
| 16 | 4 | 4 | 4.149958856 |

Table 2 presents the results of weak scaling experiment, where both the problem size and the number of processes increases at each step. The relatively stable wall- clock times across different configurations indicate good weak scaling behavior. Graph 4.0 showcases the information in the table and shows the insignificance of the increase in runtime, when both problem domain and number of process increases.



Graph 4.0 Wall-Clock Time vs Number of Nodes

The parallel efficiency is in this case calculated by the ratio of runtime of the smaller problem on 1 node and the larger problem over multiple nodes. Graph 5.0 showcases the parallel efficiency of the weak scaling experiment. As expected, the efficiency stays close to 1. This is in contrast with the strong scaling experiment and suggests that for each problem size, an optimal number of nodes exists, whereafter, increasing the number of nodes does not result in a proportional decrease in runtime.



Graph 5.0 Parallel efficiency vs Number of Nodes

**BONUS TASK**

The following report attempts to compare the Multigrid, Gauss seidel and Conjugate gradient solvers with regard to runtime and number of iterations required for convergence, solving the same problem with increasing the problem size.

Table 3 presents the results of the MultiGrid solver. This method manages to converge to the final answer with 11 iterations regardless of the problem size (number of unknowns). Table 4 illustrates the results of the Guass-Seidel solver. The method requires approximately N/2 iteration so converge, this relation would change based on the initial estimate and complexity of the problem.  Finally, table 5 represents the results of the conjugate gradient solver. It is evident that both in terms of runtime and iteration, CG solver falls between the MG and GS solvers.

*Table 3 MultiGrid Solver*

| Grid Size (Square) | Number of Iteration | Wall-Clock Time | Final Residual Norm |
|---|---|---|---|
| 17 | 11 | 0.00148128 | 7.28718e-11 |
| 33 | 11 | 0.00842667 | 3.42739e-10 |
| 65 | 11 | 0.0263147 | 5.72326e-10 |
| 129 | 11 | 0.0886188 | 8.75355e-10 |

*Table 4 Gauss-Seidel Solver*

| Grid Size (Square) | Number of Iteration | Wall-Clock Time | Final Residual Norm |
|---|---|---|---|
| 17 | 143 | 0.00829137 | 5.60281e-09 |
| 33 | 600 | 0.0304721 | 3.02503e-09 |
| 65 | 2484 | 0.124263 | 1.52608e-09 |
| 129 | 10233 | 0.465356 | 7.73458e-10 |

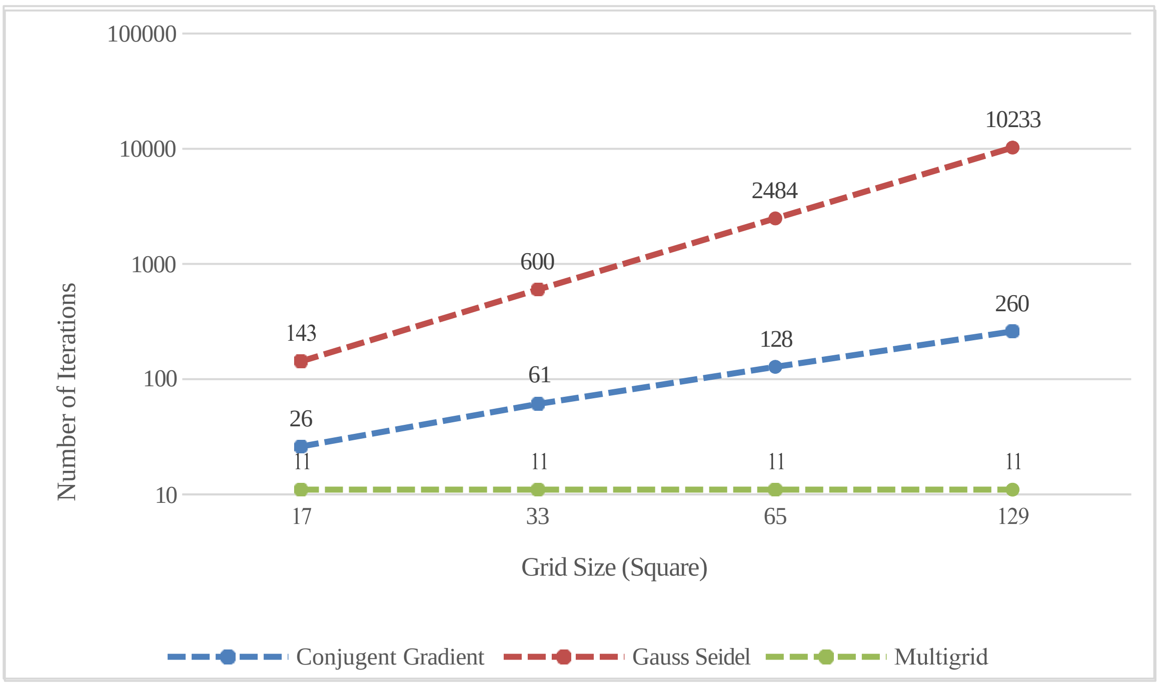
*Table 5 Conjugate Gradient solver*

| Grid Size (Square) | Number of Iteration | Wall-Clock Time | Number of MPI Processes |
|---|---|---|---|
| 17 | 25 | 0.009075219 | 1 |
| 33 | 61 | 0.022903026 | 1 |
| 65 | 128 | 0.0860997 | 1 |
| 129 | 260 | 0.107470365 | 1 |

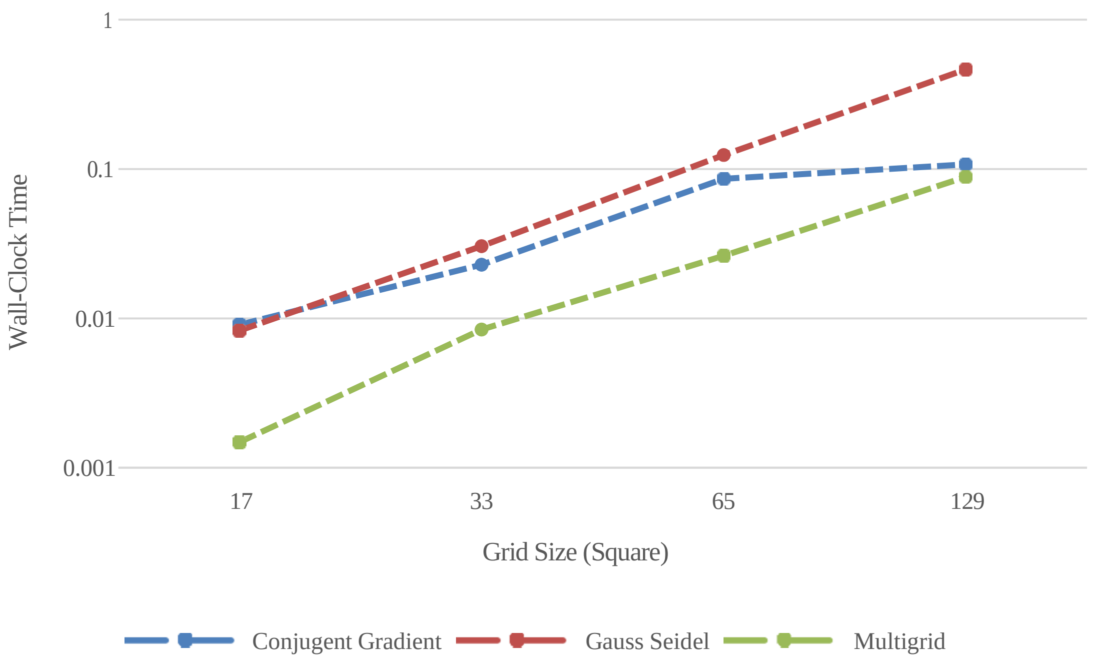
Graphs 6 compares the solvers in terms of number of iterations. It is clear that with respect to this criterion MG is by far the most efficient solver, converging with 11 iterations regardless of the problem size. CG solver also shows better scaling as the number of iterations required remains close to the sqrt of the number of unknowns in this particular problem whereas GS scales with N.

Graph 7 compares the runtime of the methods for the aforementioned domains. The quickest solver is MG, showing the efficiency of this new method. However, it is evident that for larger problem sizes, CG also improves in terms of runtime, as larger initial steps can be taken. GS shows the worst scaling in terms of runtime.

Overall it can be said that the Multigrid method required the same number of iterations regardless of the grid size, demonstrating its efficiency and scalability. Whereas, the Gauss-Seidel method required significantly more iterations as the problem size increased, highlighting its inefficiency for larger grids. The Conjugate Gradient method showed an increase in the number of iterations with grid size, but it is much more efficient than Gauss-Seidel.



Graph 6 Comparing with Number of Iterations



Graph 7 Comparing with Wall-Clock Time