CSE 570: Intro to Parallel and Distributed Programming

OpenMP - Benchmarking

Shri Harsha Adapala

UBID: 5049-5139

Hardware Specifications:

- Experiment run on UB's Vortex CCR HPC Center.
- Operating System: Ubuntu 20.04
- C++ Compiler: gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
- CPU: 8
- Memory: 32GB
- Host Name: cpn-m25

Matrix Vector Multiplication

$$T_1 = O(mn) = O(n^2)$$
$$T_p = O\left(m\frac{n}{p}\right) = O(n^2/p)$$

Size: m = n

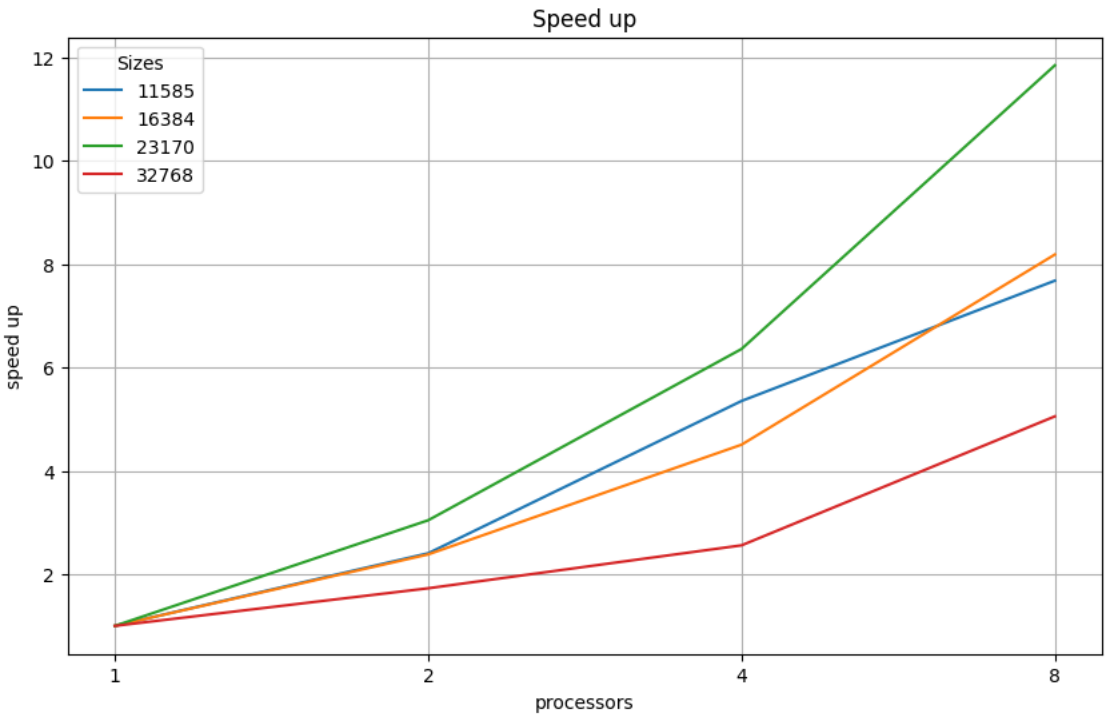
As m=n the sequential complexity is $O(n^2)$. So, we choose scaling problem size $\frac{n}{\sqrt{p}} = 11585$. So that for week scaling the diagonal elements for execution time will be constant.

Execution Time (Seconds):

Size	11585	16384	23170	32768
1	0.396196	0.795927	2.32575	4.30199
2	0.164693	0.333973	0.763955	2.49001
4	0.07401	0.176594	0.365726	1.68117
8	0.051579	0.09721	0.196278	0.851026

Speed Up:

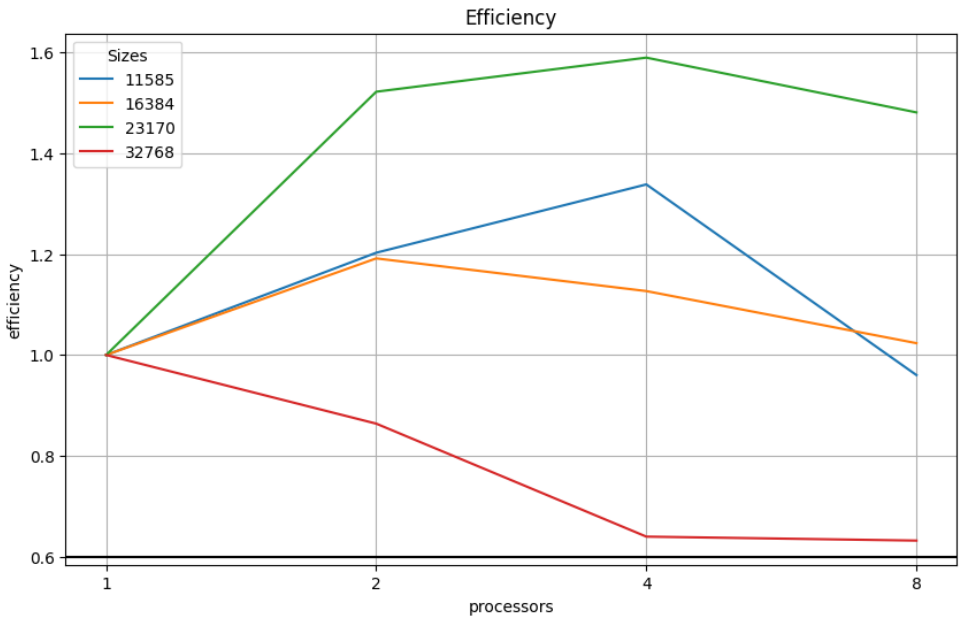
Size	11585	16384	23170	32768
1	1	1	1	1
2	2.40	2.38	3.04	1.72
4	5.35	4.50	6.35	2.55
8	7.68	8.18	11.84	5.05



The strong scaling speedup analysis shows that while parallelizing the algorithm provides significant performance improvements, the efficiency varies with problem size and number of processors. Problem sizes 16384 and 23170 demonstrate better scalability and efficiency, whereas the largest problem size 32768 indicates the presence of overhead that affects speedup gains.

Efficiency:

Size	11585	16384	23170	32768
1	1	1	1	1
2	1.20	1.19	1.522	0.86
4	1.33	1.12	1.58	0.63
8	0.96	1.02	1.48	0.63



Observations:

Strong Scaling:

Strong scaling refers to how the execution time of a fixed-size problem decreases as the number of processors increases. The data provided shows the execution time for different problem sizes as the number of processors increases from 1 to 8.

Observation:
• For smaller problem sizes (11585 and 16384), the execution time decreases significantly as the number of processors increases.
• For larger problem sizes (23170 and 32768), the execution time also decreases but not as dramatically, especially for the largest size (32768).

Conclusion:
• The system demonstrates good strong scaling for smaller problem sizes, with diminishing returns as the problem size increases. This is typical as the overhead of managing more processors can outweigh the benefits for very large problems.

Weak Scaling:

Weak scaling refers to how the execution time changes as both the problem size, and the number of processors increase proportionally.

Observation:
• The execution time does not remain constant as the problem size and the number of processors increase. For example, the execution time for size 11585 with 1 processor is 0.396196 seconds, while for size 32768 with 8 processors, it is 0.851026 seconds.

Conclusion:
• The system does not exhibit ideal weak scaling. Ideally, the execution time should remain constant if the problem size per processor remains constant. The increase in execution time suggests that the system faces challenges in maintaining efficiency as both the problem size and the number of processors increase.

Efficiency:

Efficiency measures how effectively the processors are utilized and is calculated as the speedup divided by the number of processors.

Observation:
• For smaller problem sizes (11585 and 16384), the efficiency is relatively high, especially with 2 and 4 processors.
• For larger problem sizes (23170 and 32768), the efficiency decreases, particularly with 8 processors.

Conclusion:
• The system shows good efficiency for smaller problem sizes but struggles to maintain high efficiency as the problem size and the number of processors increase. This is likely due to increased communication overhead and synchronization costs among processors.

Matrix-Matrix Multiplication

$$T_1 = O(mnk) = O(n^3)$$
$$T_p = O\left(mk\frac{n}{p}\right) = O(n^3/p)$$

Size: m = k = n

As m=n the sequential complexity is $O(n^3)$. So, we choose scaling problem size $\frac{n}{\sqrt[3]{p}} = 2048$
So that for week scaling the diagonal elements for execution time will be constant.

Execution Time (Seconds):

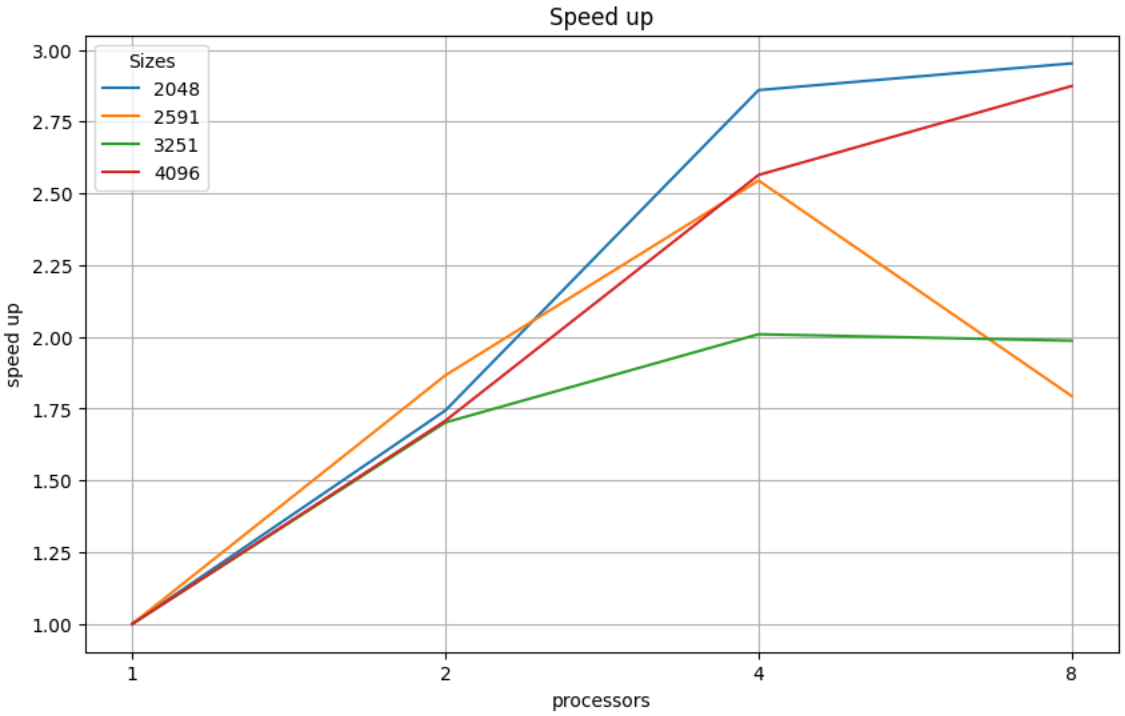
Size	2048	2581	3251	4096
1	51.1412	35.23	96.0088	412.448
2	29.3319	18.8806	56.4186	241.499
4	17.8852	13.8461	47.7907	160.875
8	17.3211	19.6421	48.3314	143.525

Million FLOPS:

Size	2048	2581	3251	4096
1	335.93	975.911	715.764	333.27
2	585.705	1821.29	1218.03	569.107
4	960.564	2483.51	1437.93	854.322
8	991.844	1750.68	1421.84	957.595

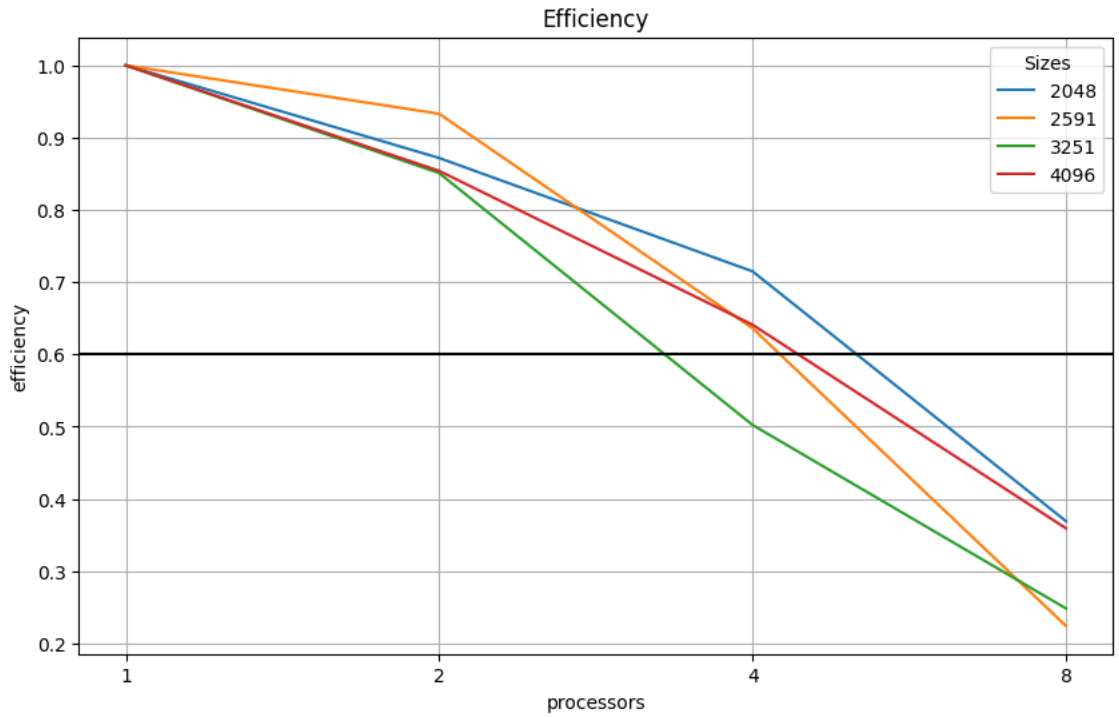
Speed Up:

Size	2048	2581	3251	4096
1	1	1	1	1
2	1.74	1.86	1.70	1.70
4	2.85	2.54	2.00	2.56
8	2.95	1.79	1.98	2.87



Efficiency:

Size	512	1024	2048	4096
1	1	1	1	1
2	0.87	0.93	0.85	0.85
4	0.71	0.63	0.50	0.64
8	0.36	0.224	0.24	0.35



Observations:

Strong Scaling:

The data shows evidence of strong scaling, as the execution time generally decreases when more processors are added for a fixed problem size. For example, for the problem size 4096:

• 1 processor: 412.448 seconds

• 2 processors: 241.499 seconds

• 4 processors: 160.875 seconds

• 8 processors: 143.525 seconds

This demonstrates that the system can solve the same problem faster with more processors, which is the essence of strong scaling.

Weak Scaling:

The data doesn't directly show weak scaling, as we don't have information about increased problem sizes with a proportional increase in processors. Weak scaling would require maintaining a constant workload per processor while increasing both problem size and processor count.

Efficiency:

The efficiency data shows a decline as the number of processors increases:

• For 2 processors, efficiency ranges from 0.85 to 0.93

• For 4 processors, efficiency ranges from 0.50 to 0.71

• For 8 processors, efficiency drops to 0.224-0.36

This decrease in efficiency is common in parallel computing and is often due to communication overhead and load imbalance.

Key observations:

1. The system shows good strong scaling up to 4 processors, with diminishing returns at 8 processors.

2. Efficiency decreases as the number of processors increases, which is typical in parallel systems due to increased communication and synchronization overhead.

3. The speedup is sublinear, meaning it doesn't increase proportionally with the number of processors. This is expected and aligns with Amdahl's Law, which states that the speedup is limited by the serial portion of the program.

4. The best efficiency is achieved with 2 processors, suggesting that this configuration might offer the best balance between performance gain and resource utilization.

5. The system's scaling behavior varies with problem size, indicating that the optimal number of processors may depend on the specific problem being solved.

In conclusion, while the system demonstrates good strong scaling characteristics, the declining efficiency with increased processor count suggests that careful consideration should be given to the optimal number of processors for each problem size to balance performance gains against resource utilization.

2D-Convolution

$$T_1 = O(mnk^2) = O(n^2)$$

$$T_p = O\left(\frac{mn}{p}k^2\right) = O(n^2/p)$$

Size: m = n; kernel size k=3.

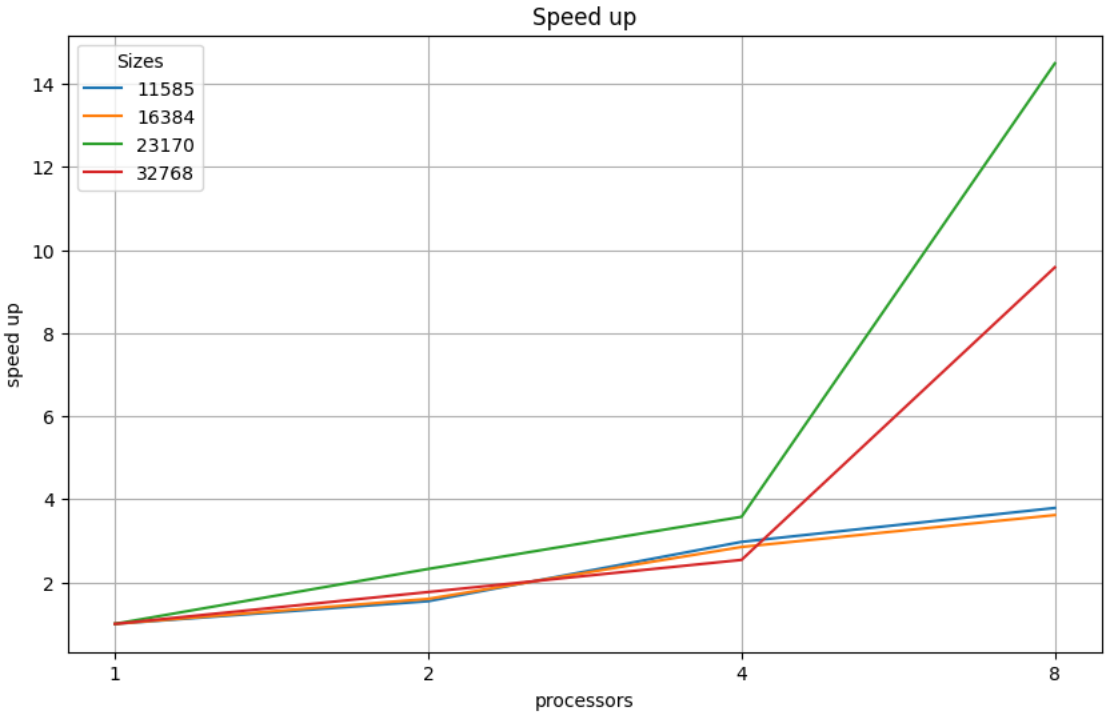
As m=n and k are constant the sequential complexity is $O(n^2)$. So, we choose scaling problem size $\frac{n}{\sqrt{p}} = 11585$. So that for week scaling the diagonal elements for execution time will be constant.

Execution Time (Seconds):

Size	11585	16384	23170	32768
1	0.496195	1.02223	2.84402	6.81005
2	0.321067	0.637996	1.22477	3.85541
4	0.166864	0.358594	0.795199	2.68135
8	0.130918	0.282495	0.196278	0.710839

Speed Up:

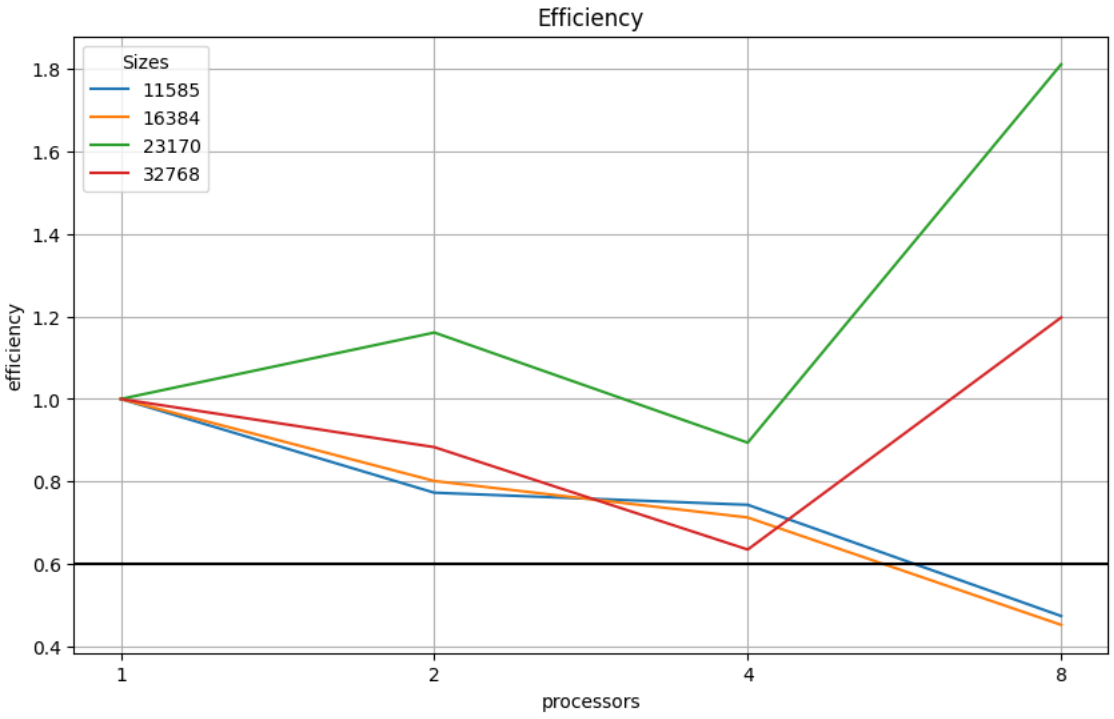
Size	11585	16384	23170	32768
1	1	1	1	1
2	1.54	1.6	2.32	1.76
4	2.97	2.85	3.57	2.53
8	3.79	3.61	14.48	9.58



The data reveals a complex relationship between problem size and processor count in strong scaling performance. Smaller problem sizes (11585 and 16384) show modest speedup improvements as processors increase, while larger sizes (23170 and 32768) demonstrate more dramatic gains, especially when scaling to 8 processors. This suggests that larger problems benefit more from increased parallelization, likely due to better computation-to-communication ratios. The most striking observation is the exceptional speedup (14.48x) achieved for the 23170-size problem with 8 processors, indicating a potential sweet spot for parallelization. However, the varied scaling efficiency across different scenarios underscores the importance of carefully matching problem size to processor count for optimal performance, considering both the computational task's nature and the underlying hardware architecture. The near-linear scaling observed for the smallest problem size (11585) when moving from 1 to 4 processors, followed by a plateau, hints at the limitations of parallelization for smaller datasets. Conversely, the superliner speedup seen in larger problems with 8 processors suggests that there might be additional factors at play, such as improved cache utilization or reduced memory contention. These insights emphasize the need for thorough benchmarking and analysis to determine the most efficient configuration for any given computational task.

Efficiency:

Size	11585	16384	23170	32768
1	1	1	1	1
2	0.77	0.80	1.16	0.88
4	0.74	0.71	0.89	0.63
8	0.47	0.45	1.81	1.97



Observations:

Strong Scaling:

Strong scaling refers to how the solution time varies with the number of processors for a fixed total problem size.

1. For all problem sizes, we see a decrease in execution time as the number of processors increases, which is expected in strong scaling.

2. The speedup is generally increasing with the number of processors, which is positive. However, the increase is not linear, indicating that there are diminishing returns as we add more processors.

3. For the largest problem size (32768), we see significant improvements in speedup, especially when moving from 4 to 8 processors (from 2.53x to 9.58x). This suggests that larger problem sizes benefit more from parallelization.

4. The most dramatic speedup is observed for size 23170 when using 8 processors, reaching a 14.48x speedup. This is an unusually high value compared to the others and might warrant further investigation.

Weak Scaling:

Weak scaling refers to how the solution time varies with the number of processors for a fixed problem size per processor. While we don't have explicit weak scaling data, we can make some observations:

1. If we look at the execution times diagonally from top-left to bottom-right, we can approximate weak scaling. For example:

- 1 processor, size 11585: 0.496195 seconds
- 2 processors, size 16384: 0.637996 seconds
- 4 processors, size 23170: 0.795199 seconds
- 8 processors, size 32768: 0.710839 seconds

2. These times are relatively close, suggesting decent weak scaling. Ideally, they would be identical for perfect weak scaling.

3. The slight increase in time from 1 to 4 processors could be due to increased communication overhead, while the decrease at 8 processors might indicate that the larger problem size allows for better utilization of the parallel resources.

Efficiency:

Efficiency is calculated as speedup divided by the number of processors, indicating how well the processors are being utilized.

1. Efficiency generally decreases as we add more processors, which is common due to increased overhead and communication costs.

2. For smaller problem sizes (11585 and 16384), efficiency drops more rapidly with increasing processors, suggesting that these problem sizes don't benefit as much from high levels of parallelization.

3. Larger problem sizes (23170 and 32768) maintain better efficiency, even showing super-linear efficiency (>1) for 8 processors. This is unusual and could be due to cache effects or other hardware-specific optimizations.

4. The highest efficiency is achieved with 2 processors across all problem sizes, indicating that this level of parallelization provides the best balance between speedup and resource utilization.

Conclusion:

This problem shows good strong scaling characteristics, especially for larger problem sizes. Weak scaling appears to be reasonable, though not perfect. Efficiency is best at lower levels of parallelization, but larger problem sizes maintain good efficiency even at higher processor counts. The unusual super-linear speedup and efficiency for some cases at 8 processors warrant further investigation to understand the underlying causes.