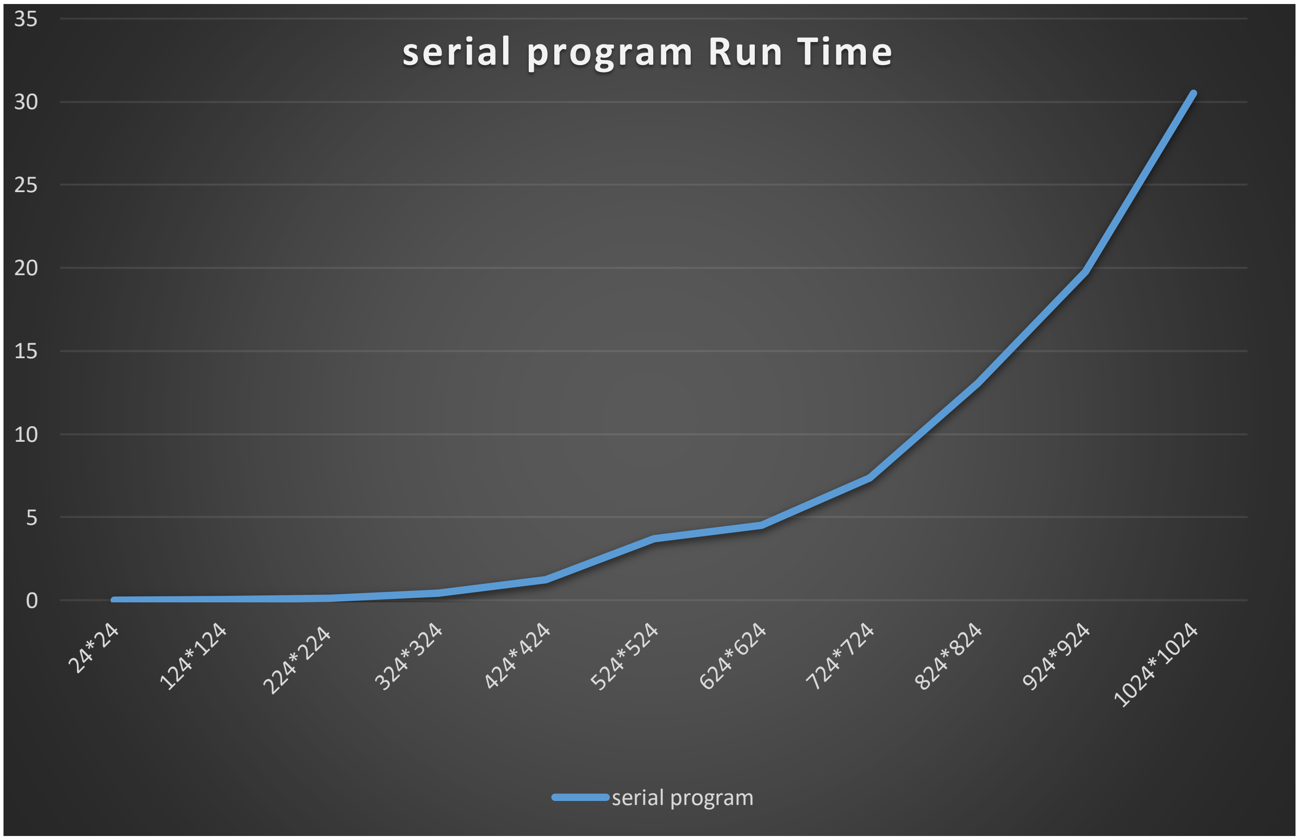
7.i. Table of serial program Run time

Size of matrices	Time taken (sec)
24*24	0.000000
124*124	0.023000
224*224	0.125000
324*324	0.436000
424*424	1.215000
524*524	3.709000
624*624	4.503000
724*724	7.359000
824*824	13.076000
924*924	19.779000
1024*1024	30.509000



7.ii. a) Table of Open MP Run time

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0.001304 s	0.002335 s	0.001658 s	0.001576 s	0.001148 s	0.0015454 s
124*124	0.012024 s	0.025004 s	0.025742 s	0.025029 s	0.025584 s	0.0256454 s
224*224	0.081041 s	0.036623 s	0.032484 s	0.036715 s	0.032544 s	0.0520621 s
324*324	0.107675 s	0.084964 s	0.062357 s	0.057841 s	0.058485 s	0.1054544 s
424*424	0.314327 s	0.216765 s	0.223587 s	0.215956 s	0.217885 s	0.2541541 s
524*524	0.841367 s	0.410568 s	0.445851 s	0.485454 s	0.448545 s	0.4551451 s
624*624	1.873978 s	0.637587 s	0.593863 s	0.552541 s	0.544544 s	0.7751154 s
724*724	2.675404 s	1.964082 s	1.667634 s	1.404032 s	1.488415 s	1.3367052 s
824*824	6.742352 s	3.554552 s	3.451425 s	3.354195 s	3.315441 s	3.4826545 s
924*924	11.64094 s	6.956692 s	5.88548 s	5.965731 s	5.584544 s	6.5415415 s
1024*1024	28.72011 s	20.88134 s	19.58164 s	19.41541 s	19.54584 s	19.785415 s

7.ii. b) Table of MPI Run time

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0.018921 s	0.009257 s	0.018335 s	0.037995 s	0.065876 s	0.058892 s
124*124	0.011169 s	0.010040 s	0.011929 s	0.055170 s	0.078321 s	0.126561 s
224*224	0.159659 s	0.033020 s	0.187613 s	0.835486 s	3.759123 s	3.525958 s
324*324	0.234009 s	0.057675 s	1.309594 s	2.707445 s	4.053291 s	5.789314 s
424*424	0.586321 s	0.124577 s	2.096765 s	5.073587 s	4.795318 s	10.91785 s
524*524	0.825414 s	0.401367 s	2.691510 s	4.458351 s	6.834815 s	12.92515 s
624*624	2.854593 s	0.639078 s	2.937587 s	6.583863 s	7.552541 s	17.75544 s
724*724	4.819804 s	0.841545 s	3.154451 s	7.345485 s	16.82525 s	23.18415 s
824*824	8.256415 s	1.806247 s	3.785463 s	9.741542 s	25.58484 s	24.652548 s
924*924	9.604454 s	3.694094 s	5.325692 s	11.36584 s	29.96573 s	24.85263 s
1024*1024	29.24696 s	5.920995s	6.852478 s	12.62016 s	32.28845 s	28.09128 s

7.iii. a) Table of Open MP Speed UP

Speedup =T_{serial}/$T_{parallel}$.

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0	0	0	0	0	0
124*124	1.912841	0.919853	0.893481	0.918934	0.898999	0.896847
224*224	1.542429	3.413156	3.848048	3.404603	3.840954	2.400979
324*324	4.049222	5.131585	6.991998	7.537906	7.454903	4.134488
424*424	3.865401	5.605148	5.434126	5.626146	5.576336	4.780564
524*524	4.408302	9.033826	8.318923	7.640271	8.268959	8.14905
624*624	2.40291	7.062566	7.582557	8.149621	8.269304	5.809458
724*724	2.750613	3.746789	4.412839	5.241334	4.944186	5.505328
824*824	1.939383	3.678663	3.78858	3.898402	3.94397	3.754607
924*924	1.69909	2.843162	3.360643	3.315436	3.541739	3.023599
1024*1024	1.062287	1.461065	1.558041	1.571381	1.560895	1.541994

7.iii. b) Table of MPI Speed UP

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0	0	0	0	0	0
124*124	2.059271	2.290837	1.928074	0.416893	0.293663	0.181731
224*224	0.782919	3.785584	0.666265	0.149614	0.033252	0.035451
324*324	1.863176	7.559601	0.332928	0.161037	0.107567	0.075311
424*424	2.072244	9.753004	0.579464	0.239476	0.253372	0.111286
524*524	4.493503	9.240919	1.378037	0.831922	0.542663	0.28696
624*624	1.577458	7.046088	1.532891	0.683945	0.596223	0.253612
724*724	1.526826	8.74463	2.332894	1.00184	0.437378	0.317415
824*824	1.583738	7.23932	3.454267	1.342293	0.511084	0.530412
924*924	2.059357	5.354222	3.713884	1.740215	0.660054	0.795851
1024*1024	1.043151	5.152681	4.452258	2.417481	0.944889	1.086067

7.iv. a) Table of Open MP Efficiency

Efficiency= SpeedUp / No. threads

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0	0	0	0	0	0
124*124	0.95642	0.229963	0.111685	0.057433	0.028094	0.014013
224*224	0.771215	0.853289	0.481006	0.212788	0.12003	0.037515
324*324	2.024611	1.282896	0.874	0.471119	0.232966	0.064601
424*424	1.932701	1.401287	0.679266	0.351634	0.174261	0.074696
524*524	2.204151	2.258457	1.039865	0.477517	0.258405	0.127329
624*624	1.201455	1.765641	0.94782	0.509351	0.258416	0.090773
724*724	1.375306	0.936697	0.551605	0.327583	0.154506	0.086021
824*824	0.969691	0.919666	0.473573	0.24365	0.123249	0.058666
924*924	0.849545	0.71079	0.42008	0.207215	0.110679	0.047244
1024*1024	0.531144	0.365266	0.194755	0.098211	0.048778	0.024094

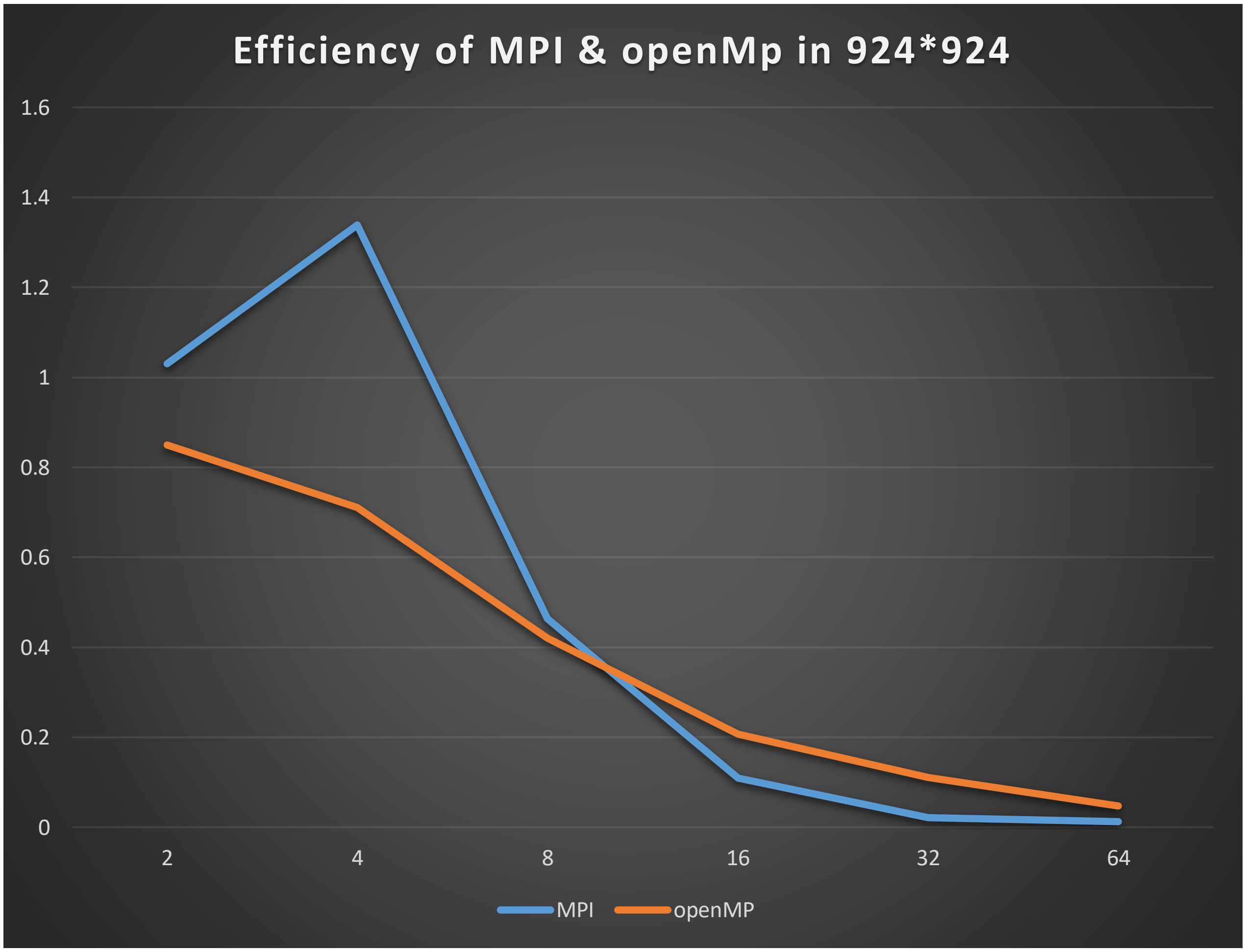
7.iv. b) Table of MPI Efficiency

Threads/ Size	2 threads	4 threads	8 threads	16 threads	32 threads	64 threads
24*24	0	0	0	0	0	0
124*124	1.029636	0.572709	0.241009	0.026056	0.009177	0.00284
224*224	0.391459	0.946396	0.083283	0.009351	0.001039	0.000554
324*324	0.931588	1.8899	0.041616	0.010065	0.003361	0.001177
424*424	1.036122	2.438251	0.072433	0.014967	0.007918	0.001739
524*524	2.246751	2.31023	0.172255	0.051995	0.016958	0.004484
624*624	0.788729	1.761522	0.191611	0.042747	0.018632	0.003963
724*724	0.763413	2.186158	0.291612	0.062615	0.013668	0.00496
824*824	0.791869	1.80983	0.431783	0.083893	0.015971	0.008288
924*924	1.029679	1.338556	0.464235	0.108763	0.020627	0.012435
1024*1024	0.521576	1.28817	0.556532	0.151093	0.029528	0.01697

7.v)Scalability

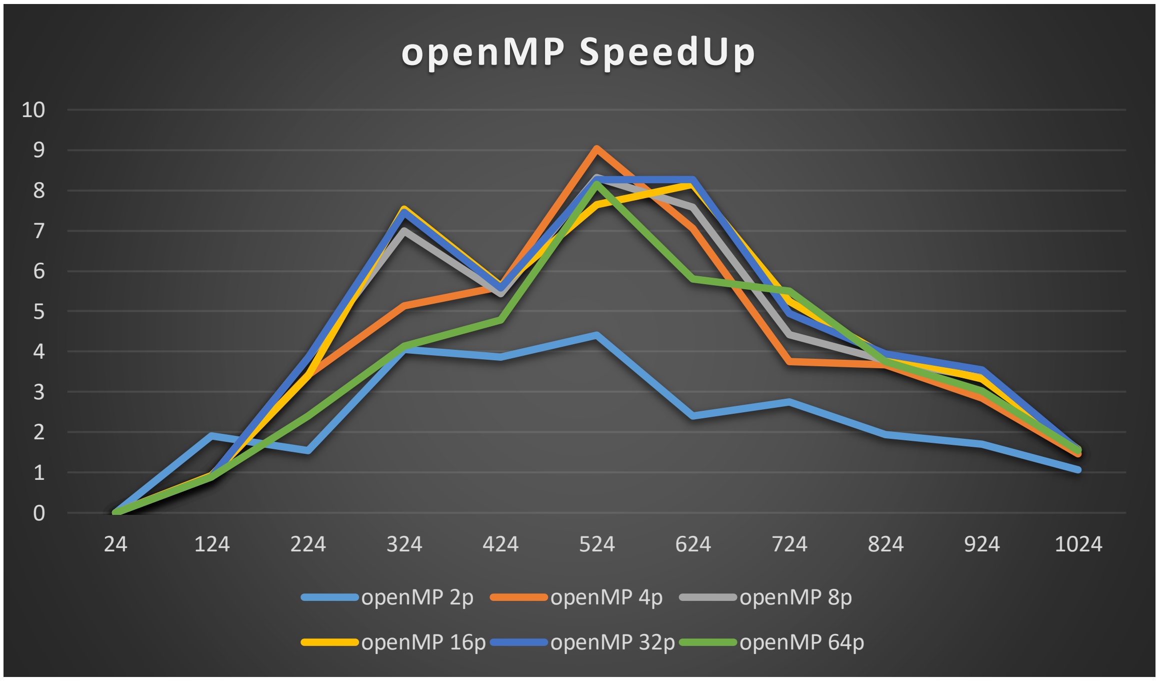
-It’s a weakly scalability as, Efficiency is not fixed with increasing number of threads in fixed size problem.

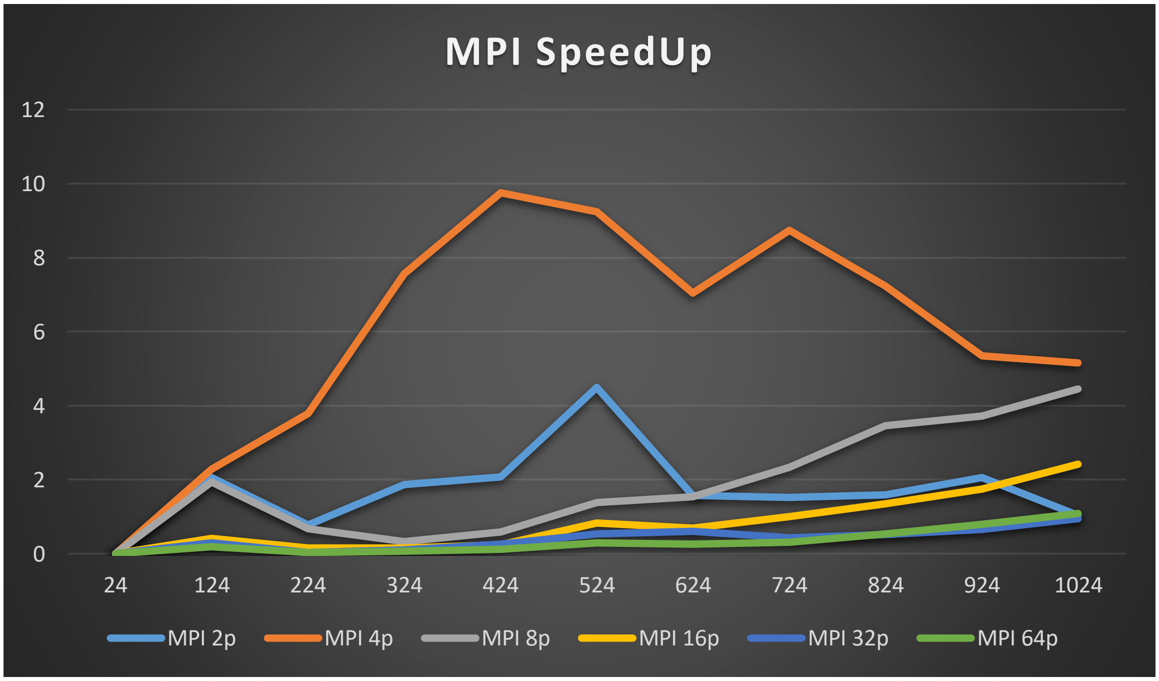
-For example this graph shows Efficiency for 924*924 matrix size of openMP and MPI programs.



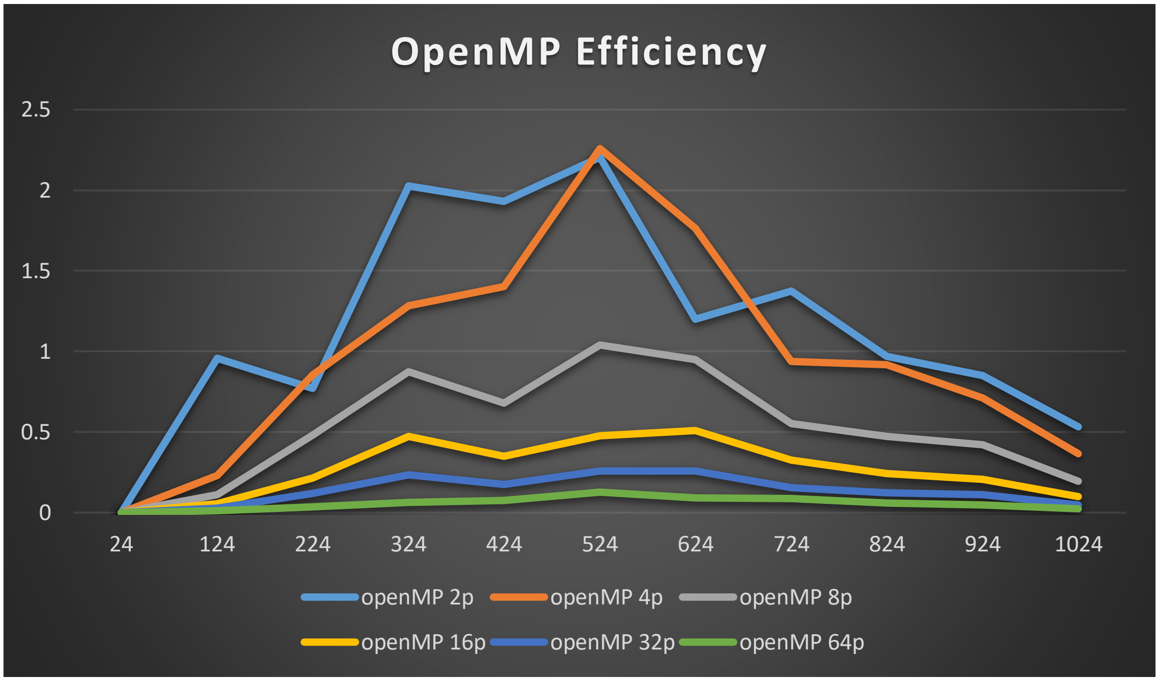
8. Graph and Comparison

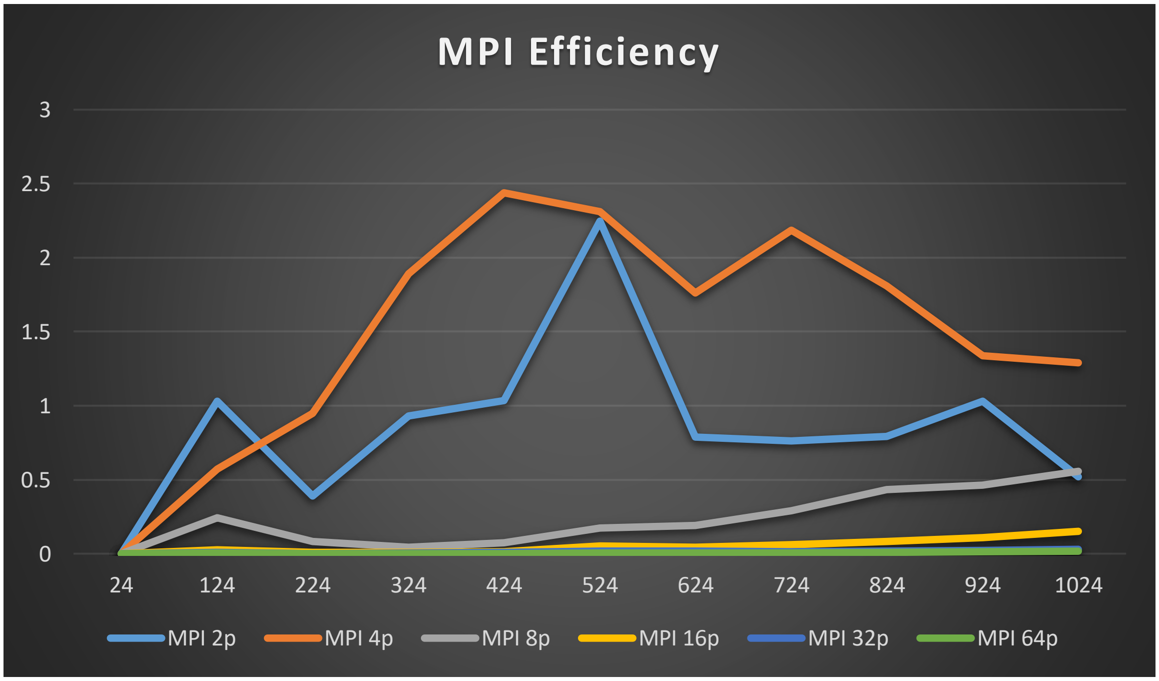
Speedup:





Efficiency:


OpenMP Efficiency
2.5
2
1.5
1
0.5
0
24
124
224
324
424
524
624
724
824
924
1024
openMP 2p
openMP 4p
openMP 8p
openMP 16p
openMP 32p
openMP 64p


MPI Efficiency
3
2.5
2
1.5
1
0.5
0
24
124
224
324
424
524
624
724
824
924
1024
MPI 2p
MPI 4p
MPI 8p
MPI 16p
MPI 32p
MPI 64p

Compare and Conclusion:

In openMP speedup graph the worst one is running program in 2 cores and the rest are close to each other.
in MPI speedup graph the more cores we use , the less speedup we get

Because they don’t share the same memory so it takes time to send data from and to cores.

But in openMp Efficiency 2 cores isn’t the worst at all.
In both efficiency graphs the more cores we use, the less efficiency we get.

From all of that we conclude that:

1. Speedup isn’t just the only measure for performance.
2. In weak scalability, using more threads/cores isn’t a guarantee for speed up.
3. In some cases there is no need to parallel code like in small input runs.
4. In MPI be careful with data because it may consume a lot of resources.
5. Foster’s methodology is best way to turn a serial program to a parallel one.
6. There is no guarantee if you run the same program and same input amount ,Run time is the same.