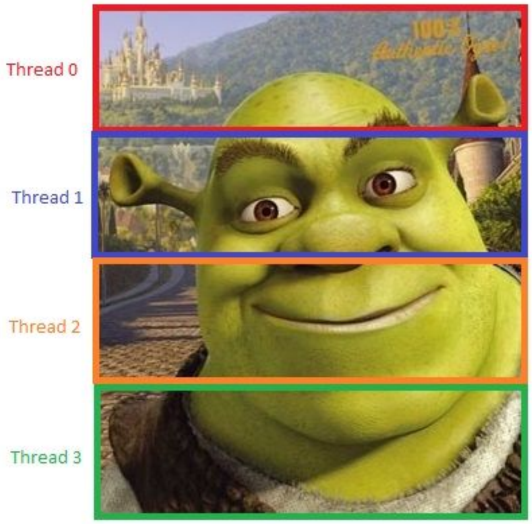
HPC_Project: MPI

Jordi Sapes
Adrià Vall-llaura

May 2016

Partitioning pattern

We used the same data partition than OpenMP, we decided to divide the matrix only in one level of decomposition. The reason for this is: For example, if we have 4 cores in the computer we can do a first level of decomposition and all the 4 cores will be used, but, if we also want to decompose the data in a second level it will be useless because the 4 threads will be occupied and the second data decomposition will have to wait for a free core. So, we think in this case it's better to only have one level of data decomposition, and with more cores the chunks of data will be smaller, so each core will have less work to do. In our case we have divided the chunks of the matrix/image like the image below:



We have **two versions of the code.** In the first one, all the processes reads the input image, this could be a problem if also they would have to write data in it but it’s not the case here. In the second version of the code, there is only one process that reads the image and then sends the chunks to other processes. With the second version there are no problems if we try to read and write in the same image but how we will see in the analysis of the performance is much much slower. Also, the second version of the code is more difficult to design and implement because we have to take in account that the kernel does not process the border of the chunk, so we have to calculate an extra size according to the size of the borders and the size of the kernel (we had a lot of problems calculating this).

Analysis and performance

SpeedUp, Efficiency and Size

As we can observe in the annex graphs the SpeedUp depends on the number of processors. As more processors, more speed up we have but the efficiency not depends on the number of processors, it depends on the quantity of the data that will be computed. As we can see when we have a small image with a small kernel, the applications is not so efficient but, when the data to compute increase, for example increasing the kernel size the efficiency grow up.

So we can conclude that when the data is higher the program has more efficiency, when data is not so big our program is not so much efficient but it has a good speed up.

Communications

Communications in MPI are so important. As we can see the program (LP), where the workers only send information to the master processor has better results than LP program, where it exists the communication in both direction. Also we can observe in the graphs and in the tables, that it is so much better communicate with smalls chunks. In the MPI with only 2 processors, in image 4, the program LS, sends big chunks and if we compare with LP it doubles the execution time. So in MPI is better to send a little amount of data.

Conclusions

In short, we have a clear conclusion to explain. If we take in account in the data of the point (2.a), we can see that having only one process that reads the image and then send it to the others clearly has a lot more time costs. In some cases it’s worse than Serial code and never is better than MPI with all the processes reading the image. With the parallel executions we can appreciate a large penalty with the 3x3 kernel, this is because the computation for each process its so small that it's insignificant versus the transmission cost. On the other side, the bigger kernels with the bigger images clearly have the best SpeedUp and Efficiency, in fact, in some cases the SpeedUp is bigger than the number of processes (for example the 99x99 kernel with im05 with 8 processes). Theoretically this is not possible but the values are near the maximum theoretical and maybe there were different external conditions in the execution of Serial code and MPI code that caused this.

Finally, we see that with this type of calculations, our data partition works really well because we have the load well balanced, if we had another type of computation depending on the specific data of every chunk, we would have to choose another data decomposition.

Annexes

Data table of MPI 8 processes with PARALLEL lecture

Img name	Image Size	Kernel Size	Serial	MPI 8 - LP	SpeedUp-LP	Efficiency-LP
Im01	1 Mb	3x3	0.013495	0.039697	0.3399501222	0.04249376527
		5x5	0.013495	0.085824	0.157240399	0.01965504987
		25x25	0.636502	0.082688	7.697634481	0.9622043102
		49x49	2.517943	0.275202	9.149435687	1.143679461
		99x99	9.583163	1.142168	8.390326992	1.048790874
Im02	2.2 Mb	3x3	0.035118	2.168095	0.01619762972	0.002024703715
		5x5	0.079011	0.028973	2.727056225	0.3408820281
		25x25	1.68588	0.207391	8.128993061	1.016124133
		49x49	6.548442	0.733102	8.932511438	1.11656393
		99x99	25.843121	2.93482	8.805692002	1.1007115
Im03	5.3 Mb	3x3	0.076702	1.930124	0.03973941571	0.004967426963
		5x5	0.158128	0.160269	0.9866412095	0.1233301512
		25x25	3.390947	0.76921	4.408350125	0.5510437657
		49x49	13.190715	1.780361	7.409011431	0.9261264289
		99x99	53.259958	6.607182	8.060918861	1.007614858
Im04	218.5 Mb	3x3	3.381978	10.332111	0.3273269132	0.04091586414
		5x5	7.864654	8.578892	0.9167447265	0.1145930908
		25x25	171.497326	22.719058	7.548610774	0.9435763468
		49x49	687.707708	87.169381	7.889326506	0.9861658132
		99x99	2810.861495	340.340161	8.258976804	1.0323721

Data table of MPI 8 processes with SERIAL lecture

img name	Image Size	Kernel Size	Serial	MPI 8 - LS	SpeedUp-LS	Efficiency-LS
Im01	1 Mb	3x3	0.013495	0.013751	0.9813831721	0.245345793
		5x5	0.013495	0.17468	0.07725555301	0.01931388825
		25x25	0.636502	0.14	4.482408451	1.120602113
		49x49	2.517943	1.3	1.936879231	0.4842198077
		99x99	9.583163	3.72	2.576119086	0.6440297715
Im02	2.2 Mb	3x3	0.035118	0.032	1.0974375	0.274359375
		5x5	0.079011	0.0467	1.691884368	0.4229710921
		25x25	1.68588	0.34	4.958470588	1.239617647
		49x49	6.548442	1.54	4.252235065	1.063058766
		99x99	25.843121	8.45	3.058357515	0.7645893787
Im03	5.3 Mb	3x3	0.076702	0.77	0.09961298701	0.02490324675
		5x5	0.158128	0.088	1.796909091	0.4492272727
		25x25	3.390947	0.627	1.284449621	0.3211124053
		49x49	13.190715	2.64	0.9742034712	0.2435508678
		99x99	53.259958	13.54	3.933527179	0.9833817947
Im04	218.5 Mb	3x3	3.381978	3.57	0.9473327731	0.2368331933
		5x5	7.864654	4.17	1.886008153	0.4715020384
		25x25	171.497326	25.62	6.693884699	1.673471175
		49x49	687.707708	95.07	7.233698412	1.808424603
		99x99	2810.861495	394.73	7.120972551	1.780243138

Data table of MPI 4 processes with SERIAL lecture

Image name	Image Size	Kernel Size	Serial	MPI 4 -LS	SpeedUp-LP	Efficiency-LP
Im01	1 Mb	3x3	0.013495	0.004441	3.038730016	0.7596825039
		5x5	0.013495	0.016663	0.8098781732	0.2024695433
		25x25	0.636502	0.160283	3.971113593	0.9927783982
		49x49	2.517943	0.647128	3.890950477	0.9727376191
		99x99	9.583163	2.649946	3.616361616	0.9040904041
Im02	2.2 Mb	3x3	0.035118	0.011574	3.034214619	0.7585536547
		5x5	0.079011	0.022317	3.540395214	0.8850988036
		25x25	1.68588	0.425051	3.966300515	0.9915751286
		49x49	6.548442	1.667568	3.92694151	0.9817353775
		99x99	25.843121	6.908857	3.740578362	0.9351445905
Im03	5.3 Mb	3x3	0.076702	0.024529	3.126992539	0.7817481349
		5x5	0.158128	0.046348	3.411754553	0.8529386381
		25x25	3.390947	0.866231	3.914598993	0.9786497482
		49x49	13.190715	3.356276	3.930163967	0.9825409919
		99x99	53.259958	13.843844	3.84719432	0.9617985799
Im04	218.5 Mb	3x3	3.381978	1.096791	3.083520926	0.7708802315
		5x5	7.864654	2.214315	3.551732251	0.8879330628
		25x25	171.497326	43.093636	3.979643908	0.9949109771
		49x49	687.707708	173.337706	3.967444383	0.9918610957
		99x99	2810.861495	697.783067	4.028274156	1.007068539

Data table of MPI 2 processes with PARALLEL lecture

Image name	Image Size	Kernel Size	Serial	MPI 2 - LP(s)	SpeedUp-LP	Efficiency-LP
Im01	1 Mb	3x3	0.013495	0.007527	0.002248727041	0.00112436352
		5x5	0.013495	0.017834	0.7567006841	0.378350342
		25x25	0.636502	0.342485	1.858481393	0.9292406967
		49x49	2.517943	1.394584	1.80551548	0.9027577399
		99x99	9.583163	6.001173	1.596881643	0.7984408215
Im02	2.2 Mb	3x3	0.035118	0.019142	1.834604535	0.9173022673
		5x5	0.079011	0.042367	1.864918451	0.9324592253
		25x25	1.68588	0.876682	1.9230234	0.9615116998
		49x49	6.548442	3.561355	1.838750139	0.9193750693
		99x99	25.843121	15.317721	1.687138772	0.8435693861
Im03	5.3 Mb	3x3	0.076702	0.042251	1.815388985	0.9076944924
		5x5	0.158128	0.082459	1.917656047	0.9588280236
		25x25	3.390947	1.741207	1.947469198	0.973734599
		49x49	13.190715	7.012564	1.88101171	0.9405058549
		99x99	53.259958	29.674217	1.794822691	0.8974113453
Im04	218.5 Mb	3x3	3.381978	1.882409	1.796622307	0.8983111534
		5x5	7.864654	4.118345	1.909663712	0.954831856
		25x25	171.497326	86.022795	1.993626527	0.9968132633
		49x49	687.707708	346.111773	1.98695266	0.99347633
		99x99	2810.861495	1529.2452	1.838071158	0.9190355788

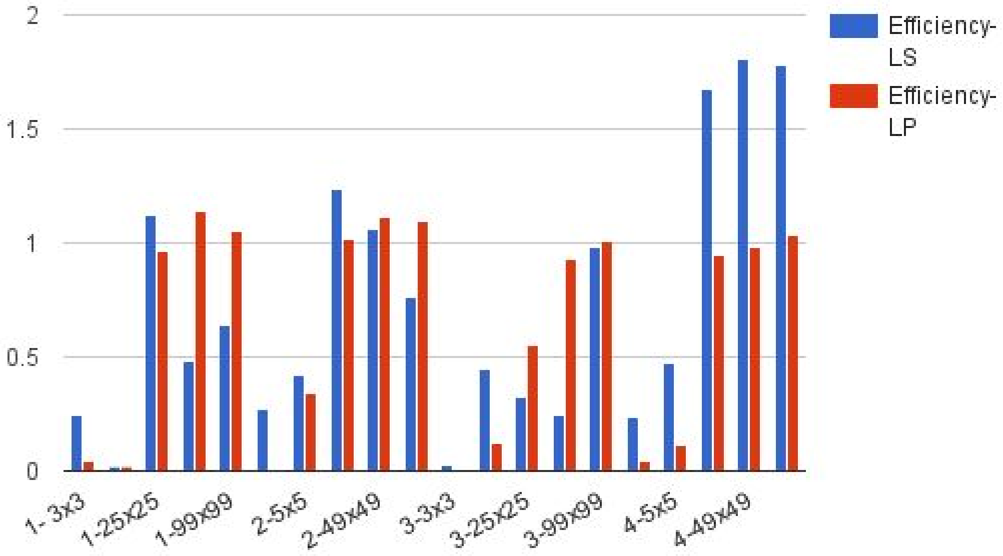
Data table of MPI 2 processes with SERIAL lecture

Image name	Image Size	Kernel Size	Serial	MPI - LS	SpeedUp-LS	Efficiency-LS
Im01	1 Mb	3x3	0.013495	0.008596	0.001176740287	0.0005883701436
		5x5	0.013495	0.017731	0.7610963849	0.3805481924
		25x25	0.636502	1	0.636502	0.318251
		49x49	2.517943	2.119837	1.187800288	0.5939001442
		99x99	9.583163	11	0.8356349746	0.4178174873
Im02	2.2 Mb	3x3	0.035118	1	0.035118	0.017559
		5x5	0.079011	0.047432	1.665774161	0.8328870805
		25x25	1.68588	1.080061	1.560911837	0.7804559187
		49x49	6.548442	5.025861	1.302949286	0.6514746429
		99x99	25.843121	26.605235	0.9713547353	0.4856773676
Im03	5.3 Mb	3x3	0.076702	0.048172	1.592252761	0.7961263805
		5x5	0.158128	0.08854	1.785949853	0.8929749266
		25x25	3.390947	2.353643	1.440722743	0.7203613717
		49x49	13.190715	8.974008	1.469880013	0.7349400067
		99x99	53.259958	44.667735	1.192358601	0.5961793003
Im04	218.5 Mb	3x3	3.381978	1	3.381978	1.690989
		5x5	7.864654	4.229334	1.859549045	0.9297745224
		25x25	171.497326	88.408961	1.939818363	0.9699091815
		49x49	687.707708	356.9875	1.926419575	0.9632097875
		99x99	2810.861495	1494.600741	1.880677172	0.940338586

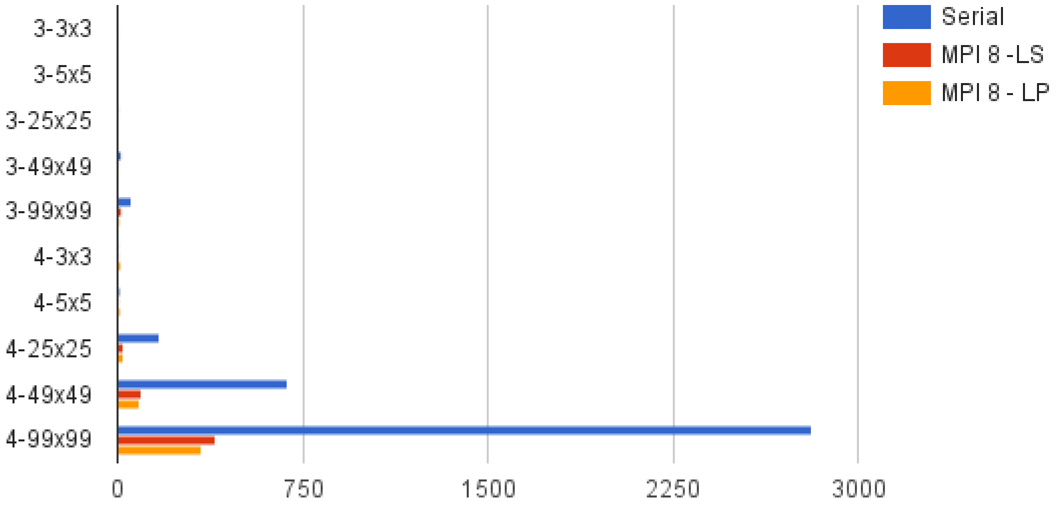
Data table of MPI 4 processes with SERIAL lecture

Image name	Image Size	Kernel Size	Serial	MPI 4 -LS	SpeedUp-LS	Efficiency-LS
Im01	1 Mb	3x3	0.013495	0.018	0.7497222222	0.1874305556
		5x5	0.013495	0.105	0.1285238095	0.03213095238
		25x25	0.636502	0.26	2.448084615	0.6120211538
		49x49	2.517943	0.96	2.622857292	0.6557143229
		99x99	9.583163	8.17	1.172969767	0.2932424419
Im02	2.2 Mb	3x3	0.035118	0.0126	2.787142857	0.6967857143
		5x5	0.079011	0.058	1.362258621	0.3405646552
		25x25	1.68588	0.62	2.71916129	0.6797903226
		49x49	6.548442	3.23	2.027381424	0.506845356
		99x99	25.843121	18.38	1.406045756	0.3515114391
Im03	5.3 Mb	3x3	0.076702	0.08	0.958775	0.23969375
		5x5	0.158128	0.105	1.505980952	0.3764952381
		25x25	3.390947	1.19	2.849535294	0.7123838235
		49x49	13.190715	5.47	2.411465265	0.6028663163
		99x99	53.259958	29.19	1.824596026	0.4561490065
Im04	218.5 Mb	3x3	3.381978	3.98	0.8497432161	0.212435804
		5x5	7.864654	5.15	1.527117282	0.3817793204
		25x25	171.497326	48.41	3.542601239	0.8856503099
		49x49	687.707708	188.54	3.647542739	0.9118856847
		99x99	2810.861495	743.65	3.779817784	0.944954446

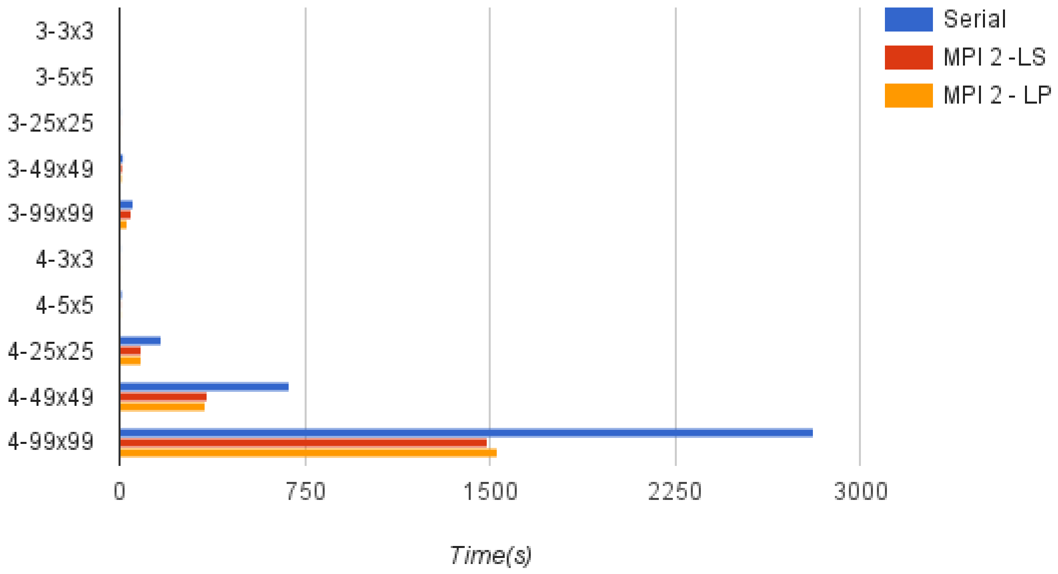
Efficiency (8 cores)



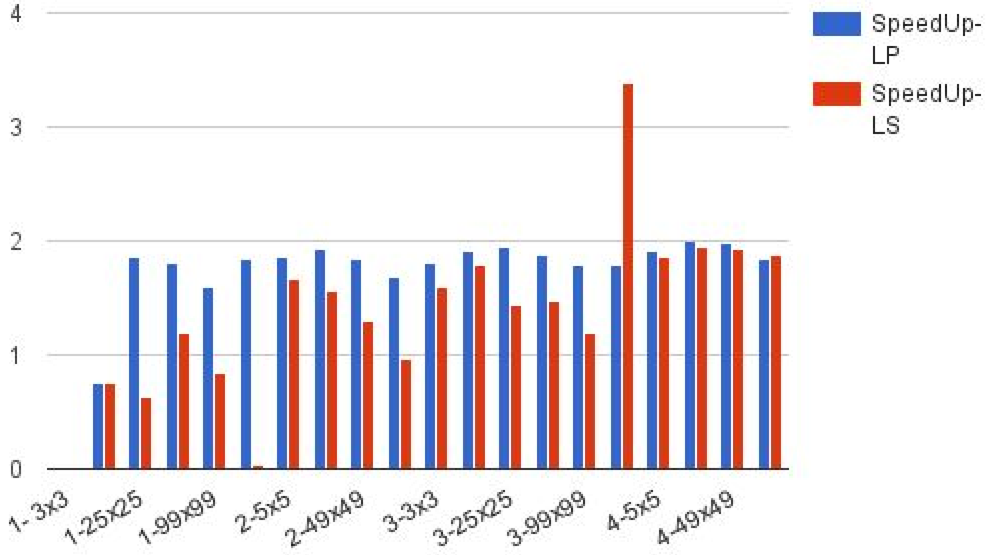
Convolution times images 3-4(8 procs)

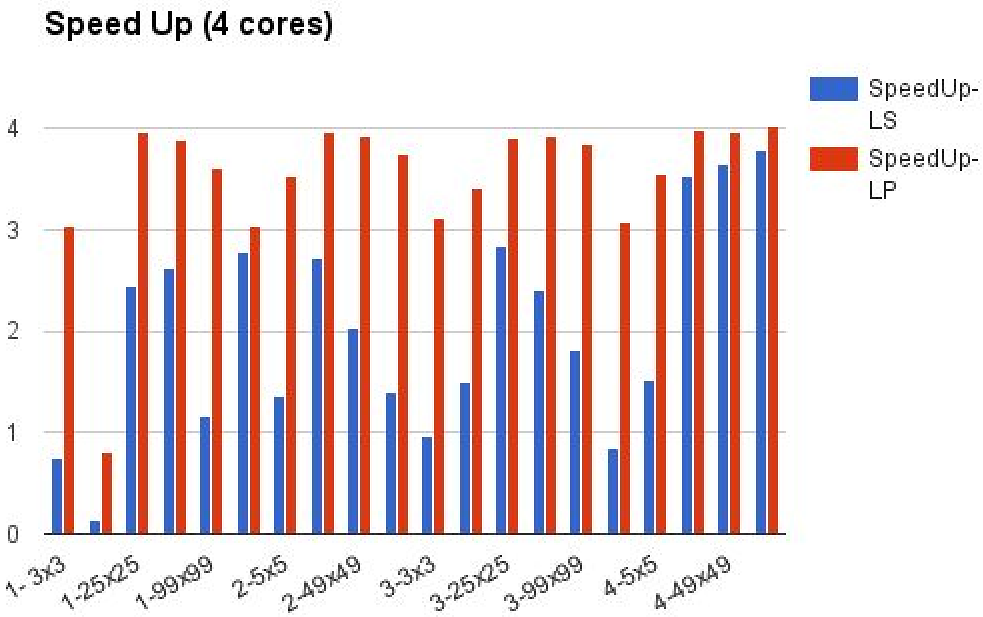


Convolution time images 3-4 (2 procesos)

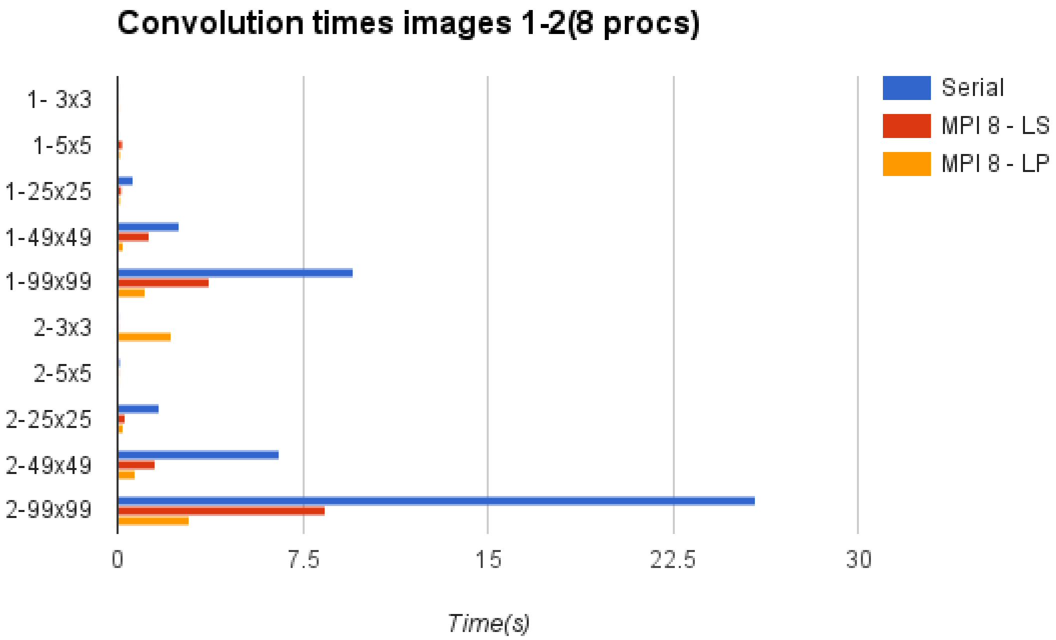


Speed-Up 2 cores

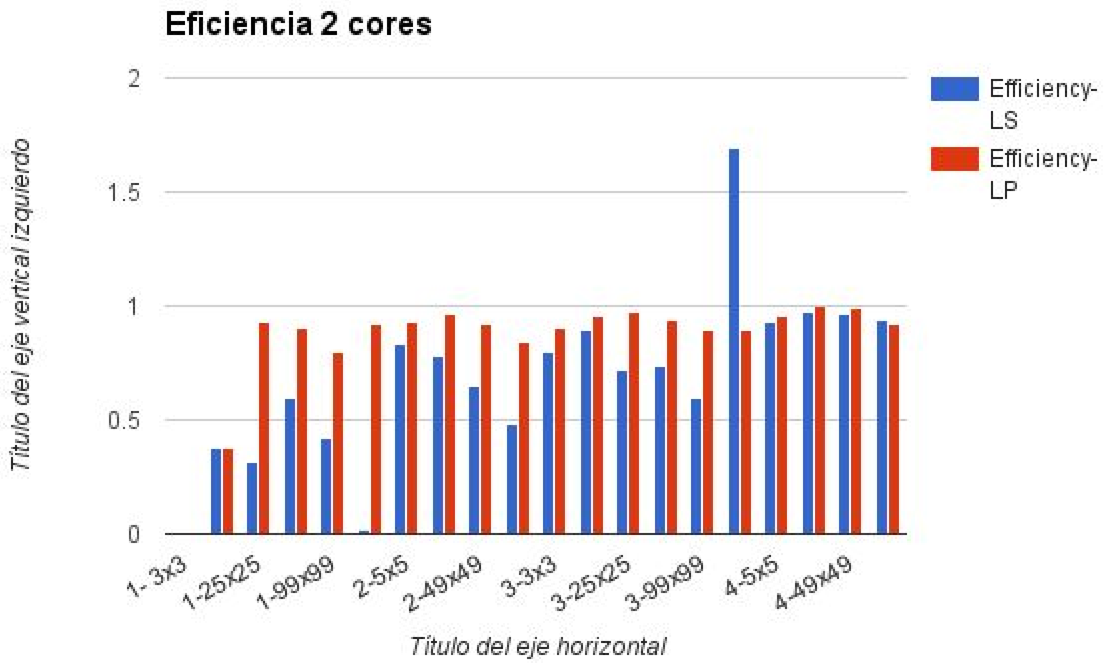


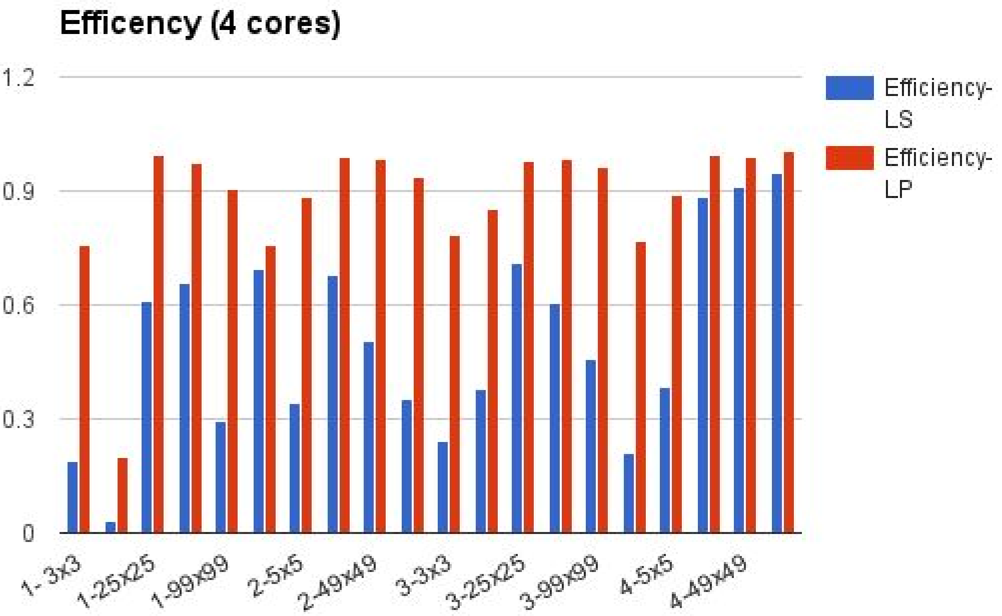


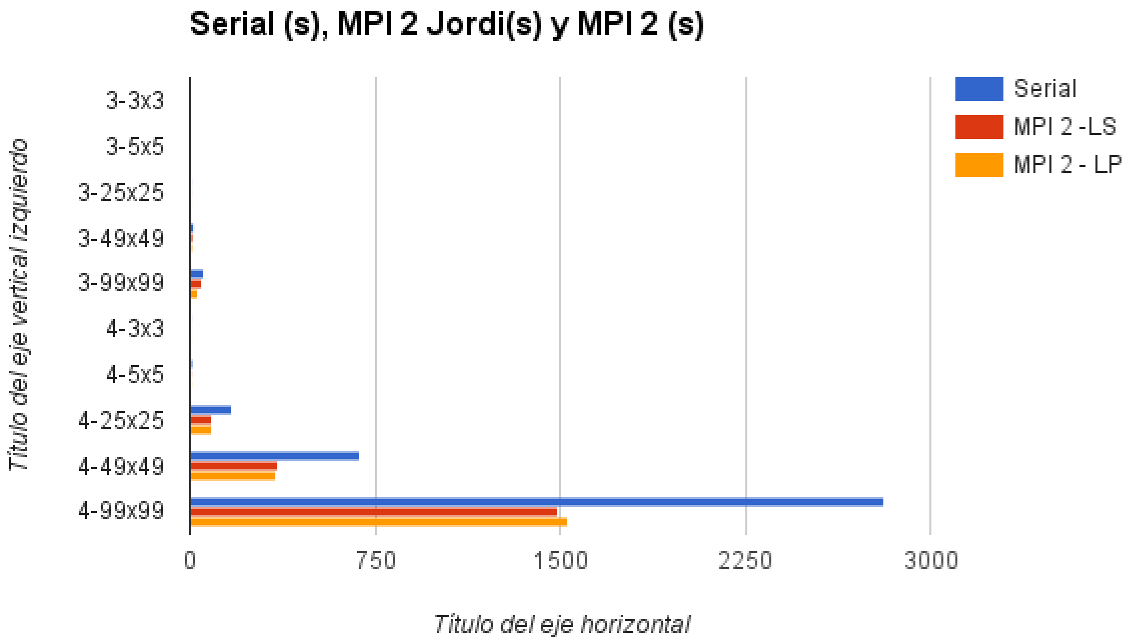
Speed Up (4 cores)

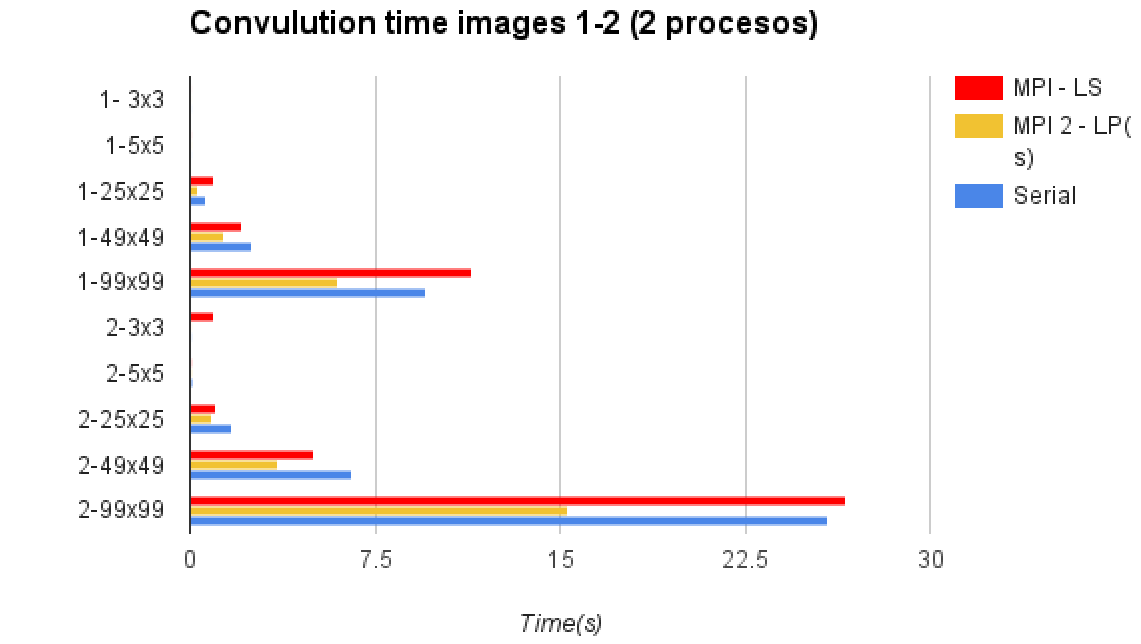


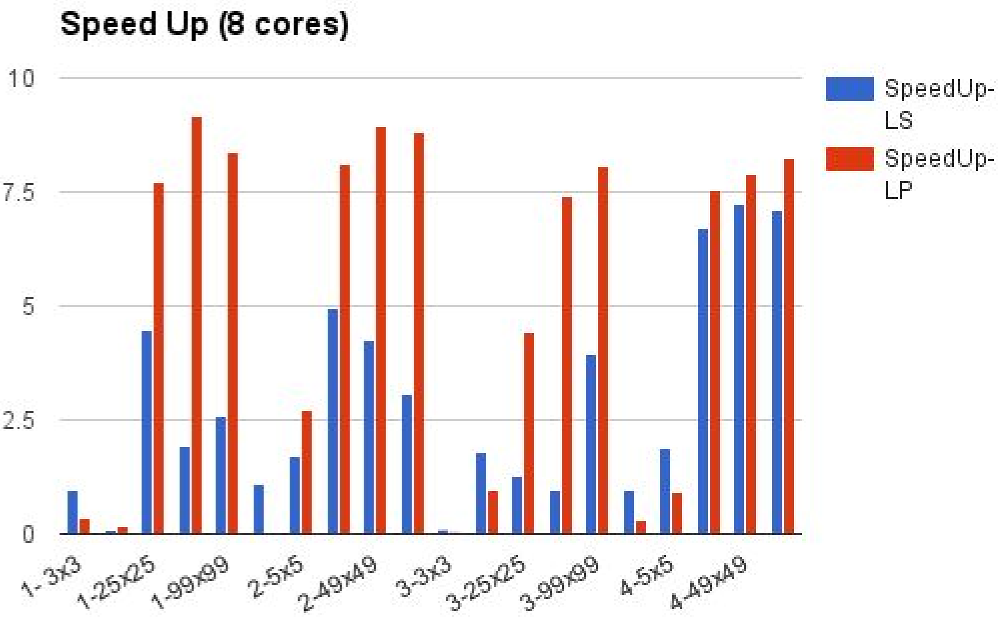
Convolution times images 1-2(8 procs)

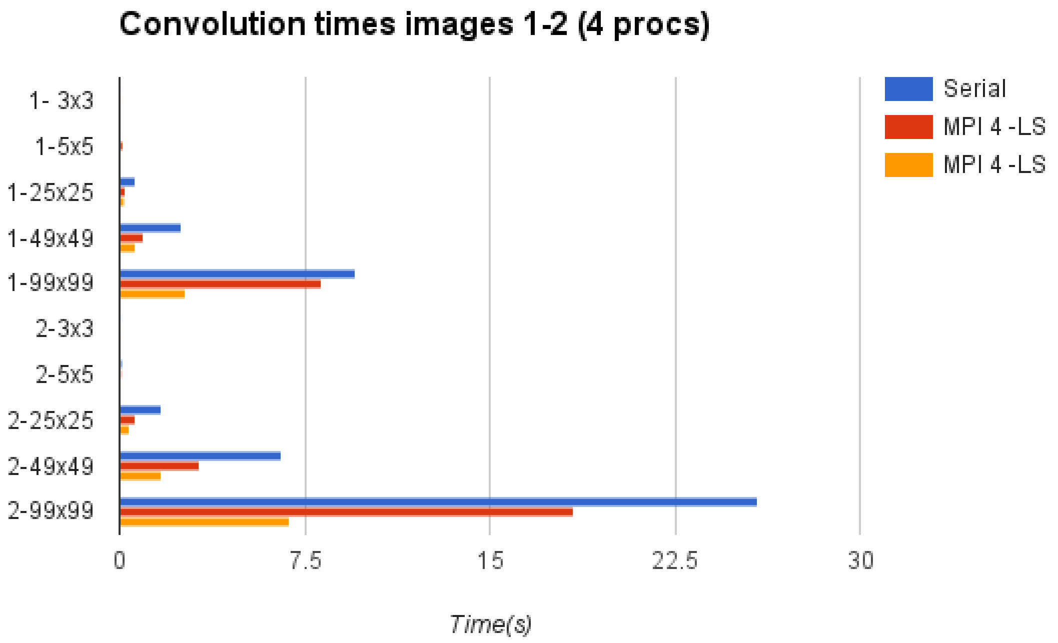

Eficiencia 2 cores
2
1.5
1
0.5
0
Título del eje vertical izquierdo
1- 3x3
1-25x25
1-99x99
2-5x5
2-49x49
3-3x3
3-25x25
3-99x99
4-5x5
4-49x49
Título del eje horizontal
Efficiency-LS
Efficiency-LP

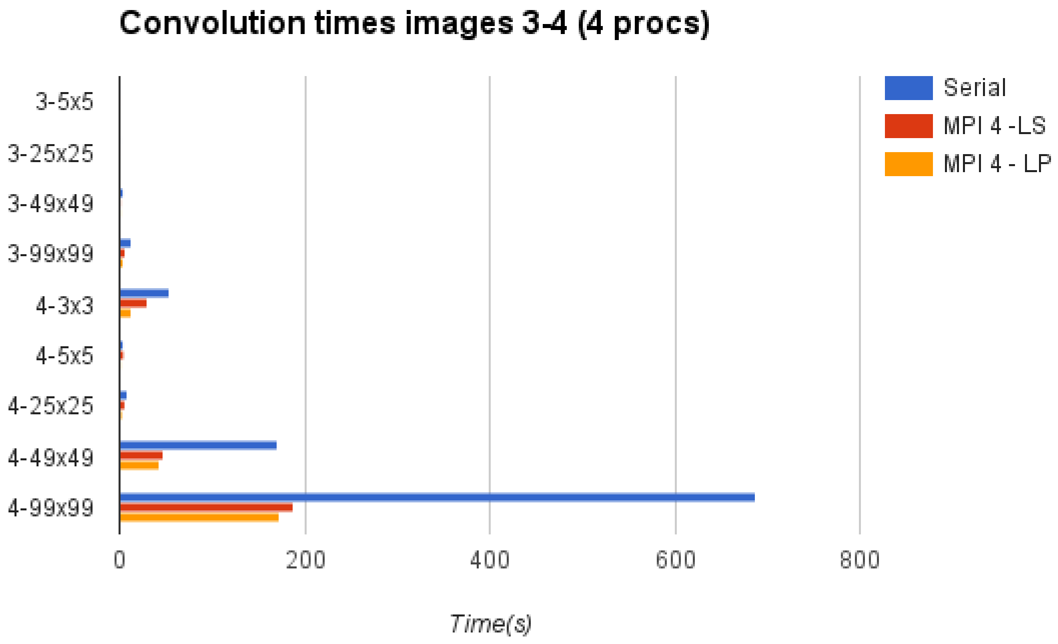

Efficency (4 cores)
1.2
0.9
0.6
0.3
0
1- 3x3
1-25x25
1-99x99
2-5x5
2-49x49
3-3x3
3-25x25
3-99x99
4-5x5
4-49x49
Efficiency-LS
Efficiency-LP


Serial (s), MPI 2 Jordi(s) y MPI 2 (s)
Serial
MPI 2 -LS
MPI 2 - LP
Título del eje vertical izquierdo
3-3x3
3-5x5
3-25x25
3-49x49
3-99x99
4-3x3
4-5x5
4-25x25
4-49x49
4-99x99
0
750
1500
2250
3000
Título del eje horizontal


Convulution time images 1-2 (2 procesos)
MPI - LS
MPI 2 - LP(s)
Serial
1- 3x3
1-5x5
1-25x25
1-49x49
1-99x99
2-3x3
2-5x5
2-25x25
2-49x49
2-99x99
0
7.5
15
22.5
30
Time(s)


Speed Up (8 cores)
10
7.5
5
2.5
0
1- 3x3
1-25x25
1-99x99
2-5x5
2-49x49
3-3x3
3-25x25
3-99x99
4-5x5
4-49x49
SpeedUp-LS
SpeedUp-LP


Convolution times images 1-2 (4 procs)
1- 3x3
1-5x5
1-25x25
1-49x49
1-99x99
2-3x3
2-5x5
2-25x25
2-49x49
2-99x99
0
7.5
15
22.5
30
Time(s)
Serial
MPI 4 -LS
MPI 4 -LS


Convolution times images 3-4 (4 procs)
3-5x5
3-25x25
3-49x49
3-99x99
4-3x3
4-5x5
4-25x25
4-49x49
4-99x99
0
200
400
600
800
Time(s)
Serial
MPI 4 -LS
MPI 4 - LP