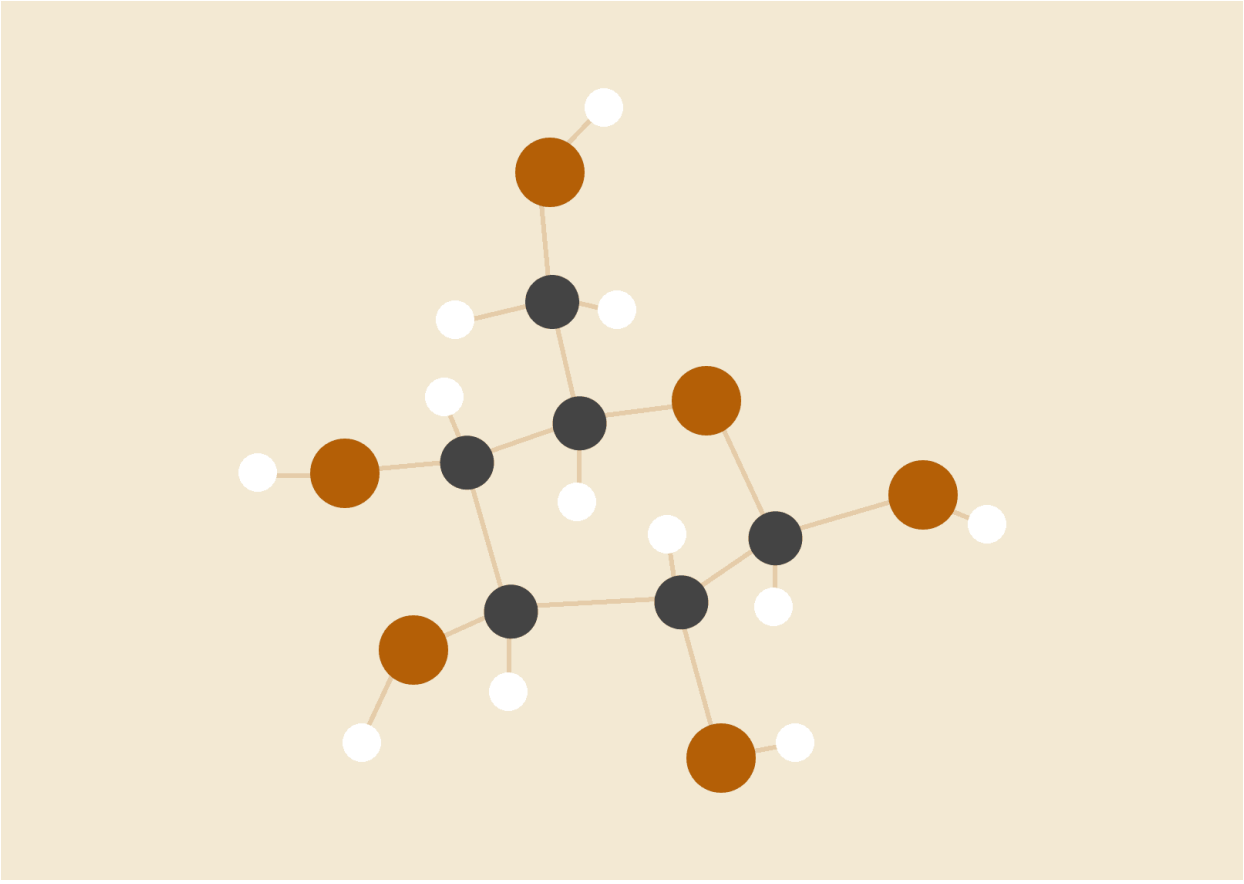
SISTEME DE OPERARE DISTRIBUITE

Studiul experimental al performanței programului

Tema 3



Mutu Gheorghiță

28.12.2022

SAI I

INTRODUCERE

Studiu experimental: studiem eficiența (i.e., performanța exprimată prin prisma timpului de execuție necesar pentru calculul fiecărei operații grafice), pe baza diverșilor factori ce influențează timpul de execuție, a programului paralel vs. cel secvențial (OpenMP & MPI).

Tema 3 este o continuare a temei 1 -- adaptăm aplicația paralelă de la tema 1 (cea care rula distribuit pe un cluster, folosind comunicații prin API-ul MPI), înlocuind task-urile secvențiale din care este format job-ul MPI, cu procese *multi-threaded*.

Utilizăm OpenMP pentru a "transforma" procesele *single-threaded* ce alcătuiesc job-ul MPI de la tema 1, în procese *multi-threaded.*

PROCEDURA

Variația timpului de execuție a calculului pentru fiecare dintre operațiile grafice implementate, în funcție de următorii parametri:

- dimensiunile n*m (i.e. rezoluția) imaginii input (recomandare: folosiți ca sample-uri imagini cu rezoluție mare, i.e. zeci de megapixeli);
- numărul de procese alocat pentru execuția job-ului MPI (respectiv, de observat limitarea dată de numărul de "procesoare"/core-uri disponibile pe cele două calculatoare ce formează clusterul configurat pe care veți testa programul paralel);
- modul de alegere a "divizării" matricii imaginii inițiale în submatrici: divizare 1D (i.e., în "felii" orizontale sau verticale) sau divizare 2D Show / Hide the details
- respectiv modul de trimitere a submatricilor spre prelucrare de către procesele worker: load-balancing static sau dinamic.

Alți factori/parametri ce ar putea influența timpul de execuție:

- caracteristicile hardware ale mașinilor clusterului pe care se rulează programul paralel (i.e., numărul de core-uri HW, cu HT sau nu; rulat direct pe mașina gazdă sau într-o mașină virtuală, etc.);
- gestiunea memoriei virtuale (i.e. rata erorilor de pagină);
- containere Docker (și orchestratoare pentru acestea) pentru un control mai fin al

resurselor gazdei alocate pentru execuția programului;

Suplimentar trebuie adăugat și un pas de paralelizare a buclelor for din implementările fiecăreia dintre cele 3 operații grafice alese la realizarea temei 1.

Pentru fiecare operație grafică, se vor folosi concluziile trase din realizarea temei 2 referitoare la modul optim de paralelizare a buclelor for (i.e., câte bucle for să paralelizăm, care anume dintre ele și în ce mod anume -- cu collapse sau cu nested), și respectiv modul de alegere a 'schedule'-ului (i.e., "împărțirea iterațiilor în chunk-uri") pentru buclele for paralelizate.

De asemenea, modul optim de paralelizare poate fi decis și pe baze experimentale (folosind imagini cu rezoluții de ordinul zeci de megapixeli, i.e. mărimea obișnuită a imaginilor produse de modelele uzuale de aparate foto și telefoane disponibile în prezent), pentru a obține cea mai eficientă implementare paralelizată a operației grafice respective.

PARAMETRI

PARAMETRU	VALORI POSIBILE
Metoda de execuție	Secvențială Paralelă
Dimensiuni matrice	n x m
Număr de procese	1-12
Tipul divizării	1-2D
Load-Balancing	Static Dinamic
Communication type	Scatter/Gather Send/Receive
OpenMP for loop parallelization	V #0 - #17 from Homework 02

INPUT

Fișier	Pixels	Coloane	Rânduri	Canale
example01.jpg	47619036	3254	4878	3
example02.jpg	136323072	5504	8256	3
example03.jpg	96238398	5494	5839	3
example04.jpg	218131305	10315	7049	3

REZULTATE

Toate rezultatele sunt calculate pe binare optimizate cu -O3.
Acțiunile posibile sunt:

- 0 - `ACTION_COLOR_INVERSION`
- 1 - `ACTION_BLUE_AND_RED_SWITCH`
- 2 - `ACTION_GAUSSIAN_BLUR`
- 3 - `ACTION_VERTICAL_FLIP`

Pentru 0, 1 avem Scatter & Gather. Pentru 2, 3 avem Send & Receive.

Toate rezultatele sunt media aritmetică a 5 rulări consecutive ale unei acțiuni.

Numărul de threads pentru OpenMP pe WSL este de 12 (6x2).

WSL2 instanță specificații

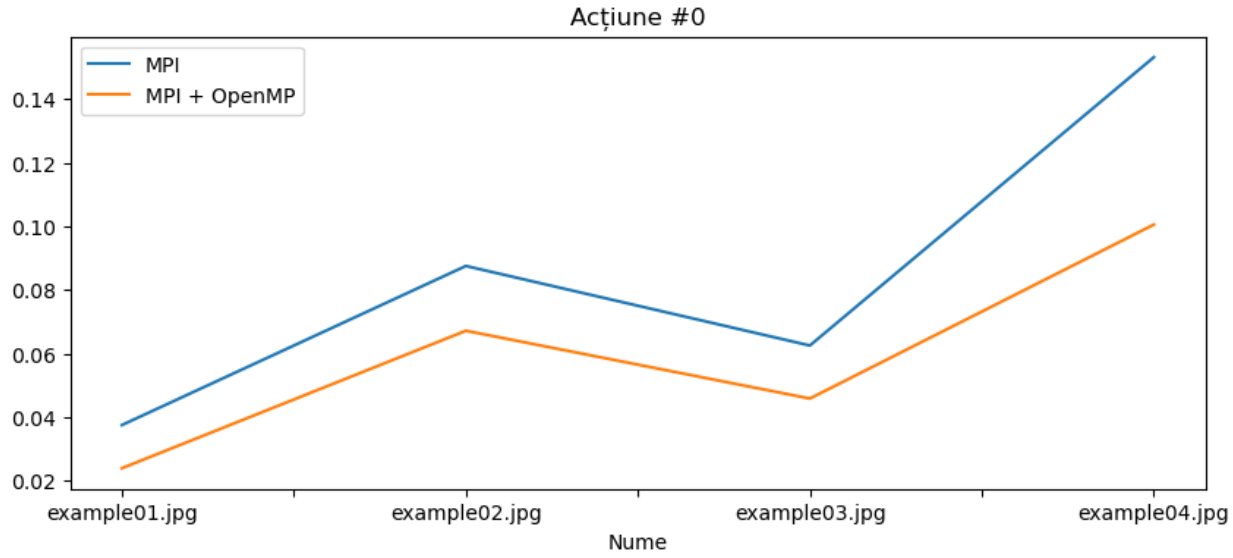
Spec	Value
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
Address sizes	36 bits physical, 48 bits virtual
CPU(s)	12
On-line CPU(s) list	0-11
Thread(s) per core	2
Core(s) per socket	6
Socket(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	165
Model name	Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz
Stepping	2
CPU MHz	2712.000

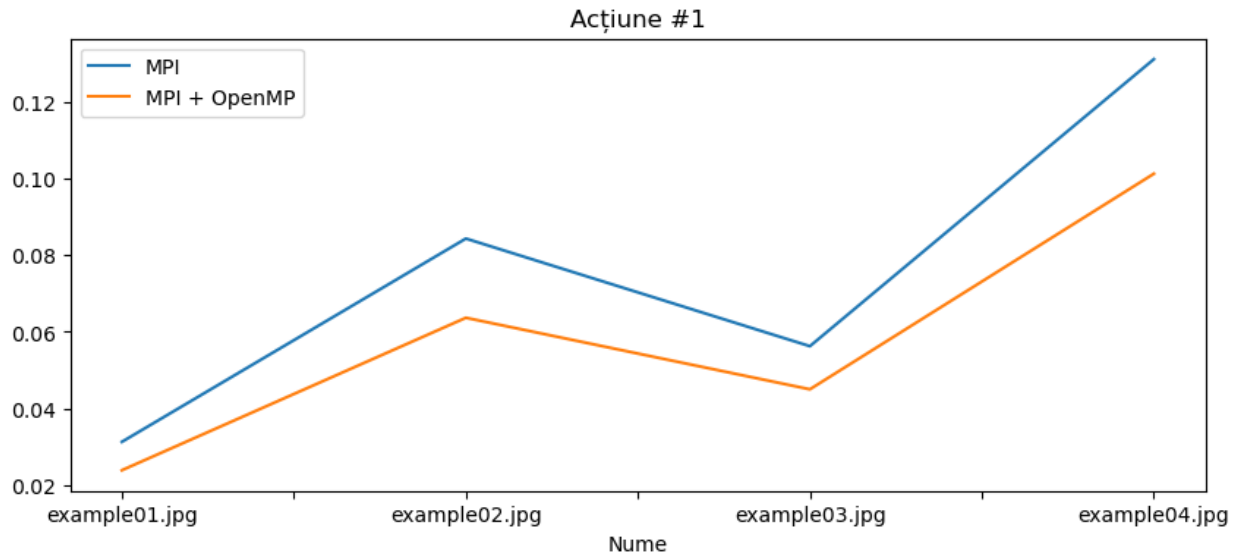
CPU max MHz	2712.000
BogoMIPS	5424.00
Hypervisor vendor	Windows Subsystem for Linux
Virtualization type	container
Flags	fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes 64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave osxsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch fsgsbase tsc_adjust b mi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

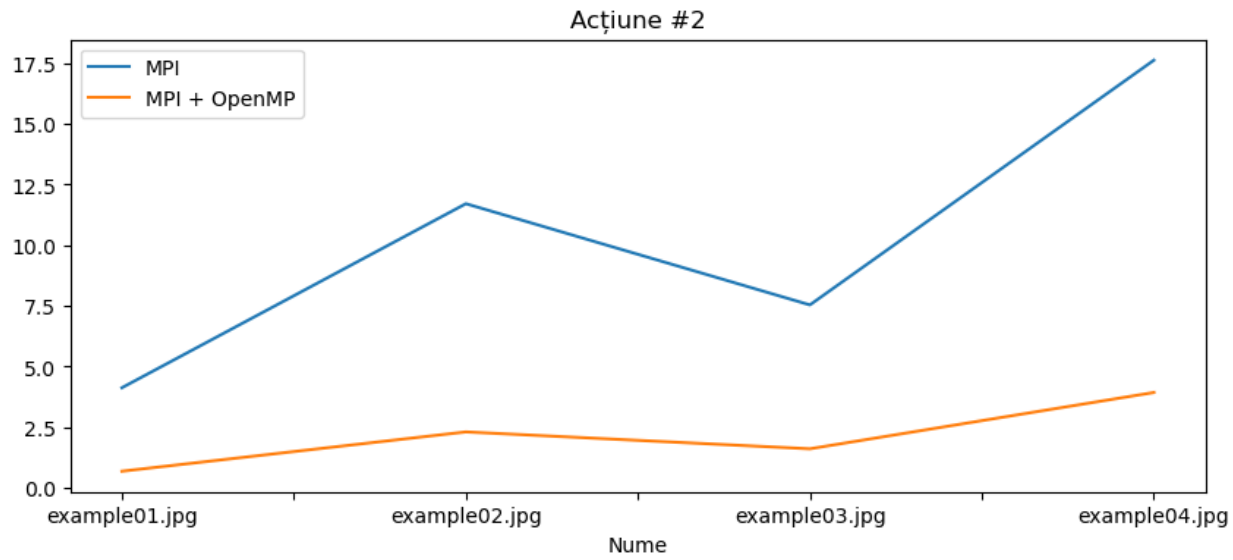
Execuție secvențială (WSL2)

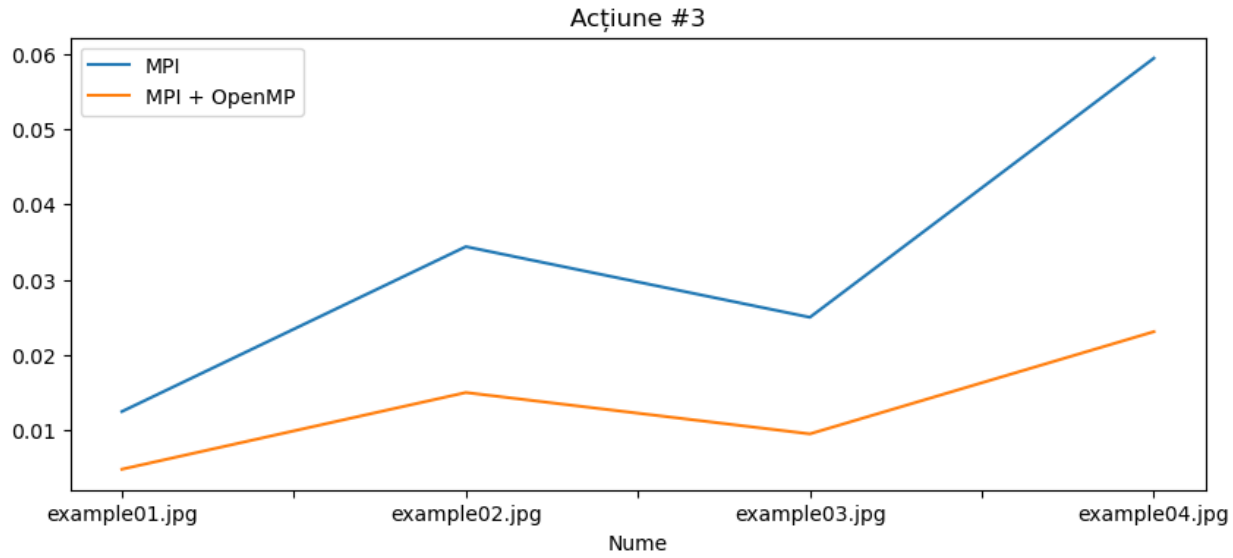
Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	1	secvențială	0	0.037500	0.023928
example01.jpg	3254	4878	1	secvențială	1	0.031250	0.023835
example01.jpg	3254	4878	1	secvențială	2 (r5)	4.131250	0.692215
example01.jpg	3254	4878	1	secvențială	3	0.012500	0.004844
example02.jpg	5504	8256	1	secvențială	0	0.087500	0.067142
example02.jpg	5504	8256	1	secvențială	1	0.084375	0.063673
example02.jpg	5504	8256	1	secvențială	2 (r5)	11.712500	2.311897
example02.jpg	5504	8256	1	secvențială	3	0.034375	0.015018
example03.jpg	5494	5830	1	secvențială	0	0.062500	0.045833
example03.jpg	5494	5830	1	secvențială	1	0.056250	0.044984
example03.jpg	5494	5830	1	secvențială	2 (r5)	7.543750	1.616268
example03.jpg	5494	5830	1	secvențială	3	0.025000	0.009534
example04.jpg	10315	7040	1	secvențială	0	0.153125	0.100516
example04.jpg	10315	7040	1	secvențială	1	0.131250	0.101320
example04.jpg	10315	7040	1	secvențială	2 (r5)	17.625000	3.935946

example04.jpg	10315	7040	1	secvențială	3	0.059375	0.023094





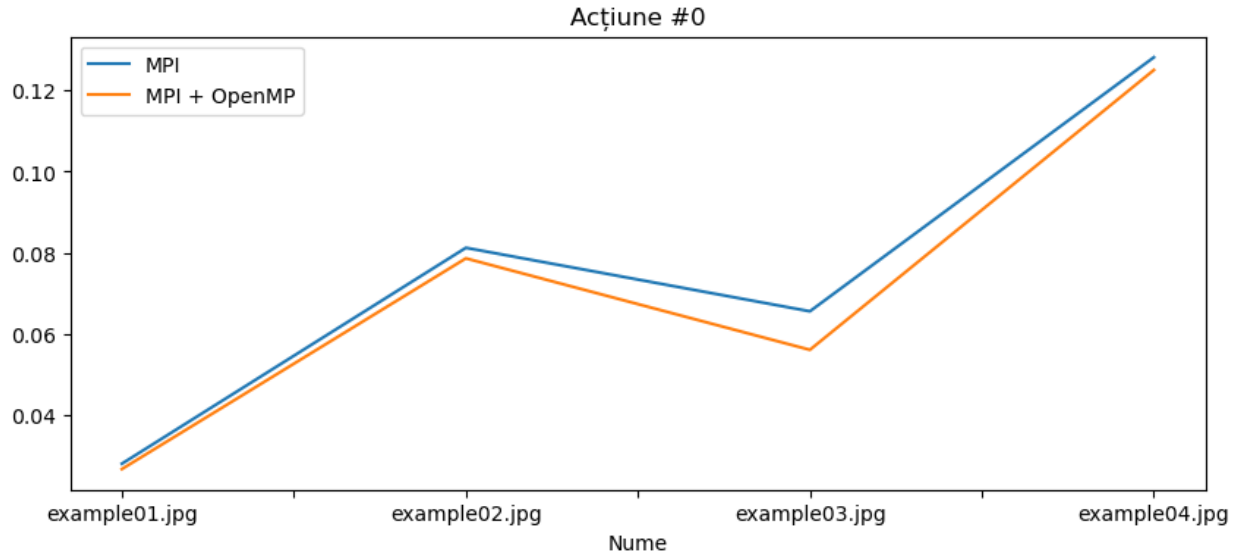

Acțiune #2
MPI
MPI + OpenMP
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

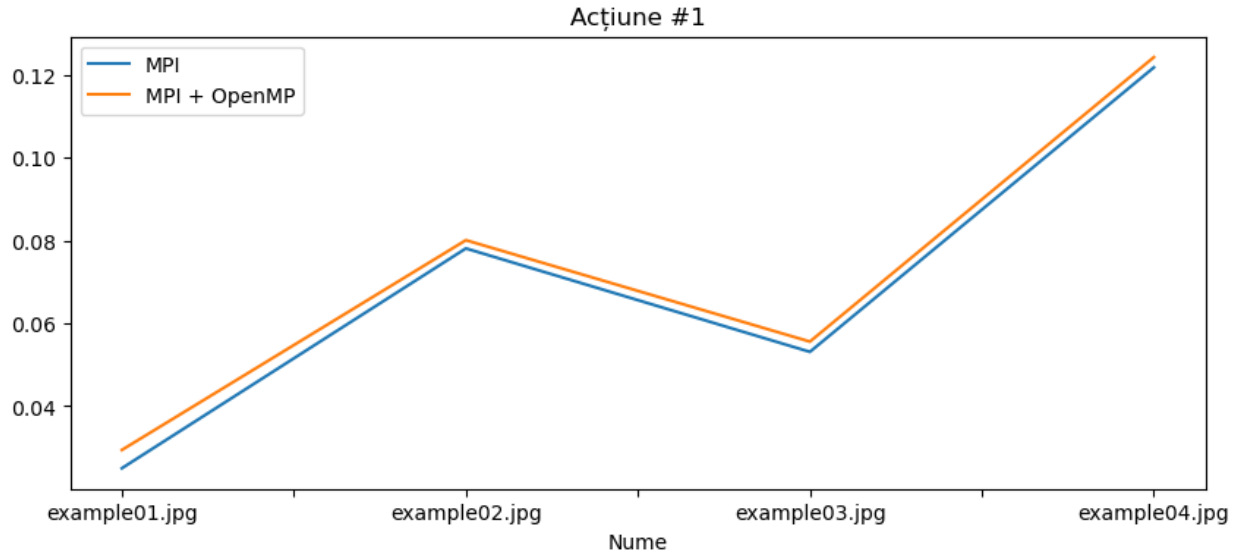

Acțiune #3
MPI
MPI + OpenMP
0.06
0.05
0.04
0.03
0.02
0.01
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

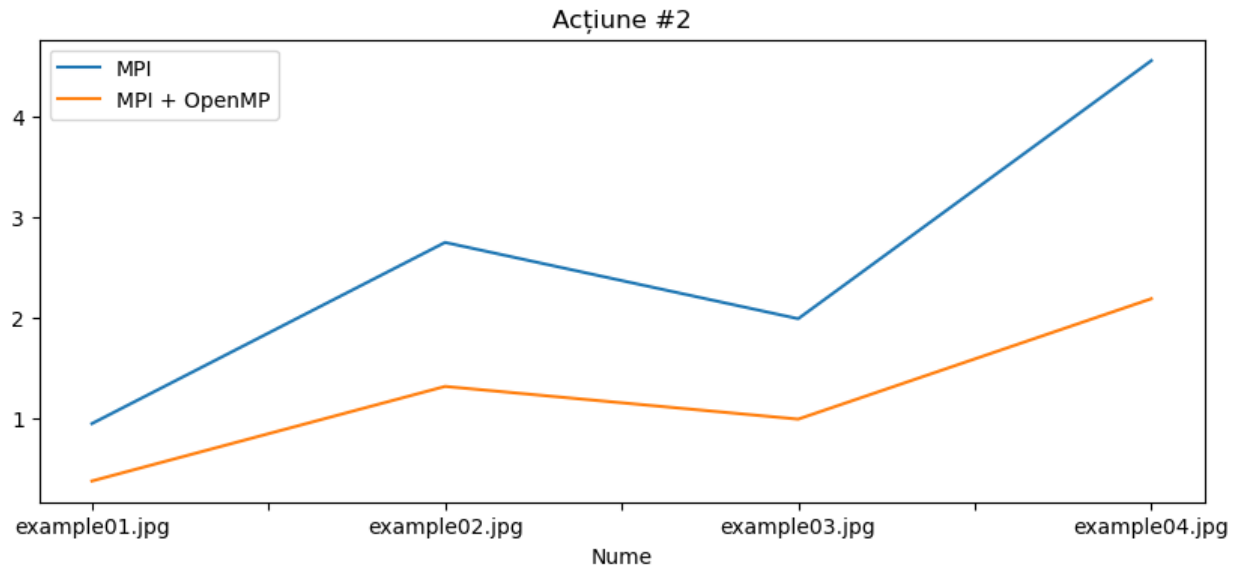
Execuție paralelă (#2) (WSL2) pe aceeași mașină

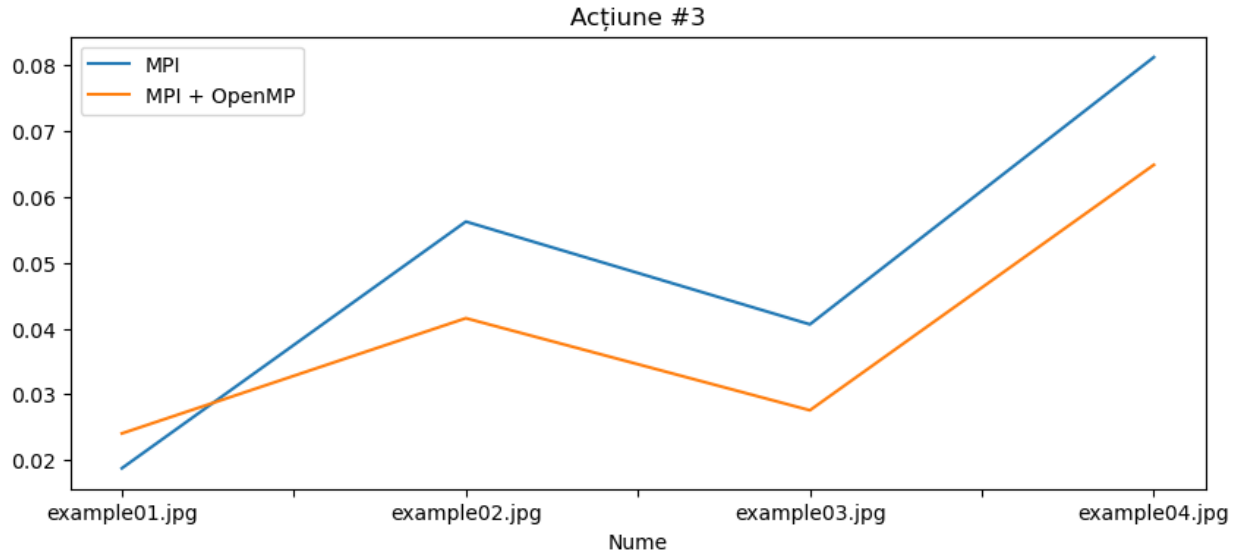
Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	2	paralelă	0	0.028125	0.026805
example01.jpg	3254	4878	2	paralelă	1	0.025000	0.029417
example01.jpg	3254	4878	2	paralelă	2 (r5)	0.943750	0.372347
example01.jpg	3254	4878	2	paralelă	3	0.018750	0.024040
example02.jpg	5504	8256	2	paralelă	0	0.081250	0.078650
example02.jpg	5504	8256	2	paralelă	1	0.078125	0.080118
example02.jpg	5504	8256	2	paralelă	2 (r5)	2.746875	1.313654
example02.jpg	5504	8256	2	paralelă	3	0.056250	0.041560
example03.jpg	5494	5830	2	paralelă	0	0.065625	0.056145
example03.jpg	5494	5830	2	paralelă	1	0.053125	0.055600
example03.jpg	5494	5830	2	paralelă	2 (r5)	1.987500	0.988058
example03.jpg	5494	5830	2	paralelă	3	0.040625	0.027563
example04.jpg	10315	7040	2	paralelă	0	0.128125	0.124997
example04.jpg	10315	7040	2	paralelă	1	0.121875	0.124345
example04.jpg	10315	7040	2	paralelă	2 (r5)	4.556250	2.186234

example04.jpg	10315	7040	2	paralelă	3	0.081250	0.064884





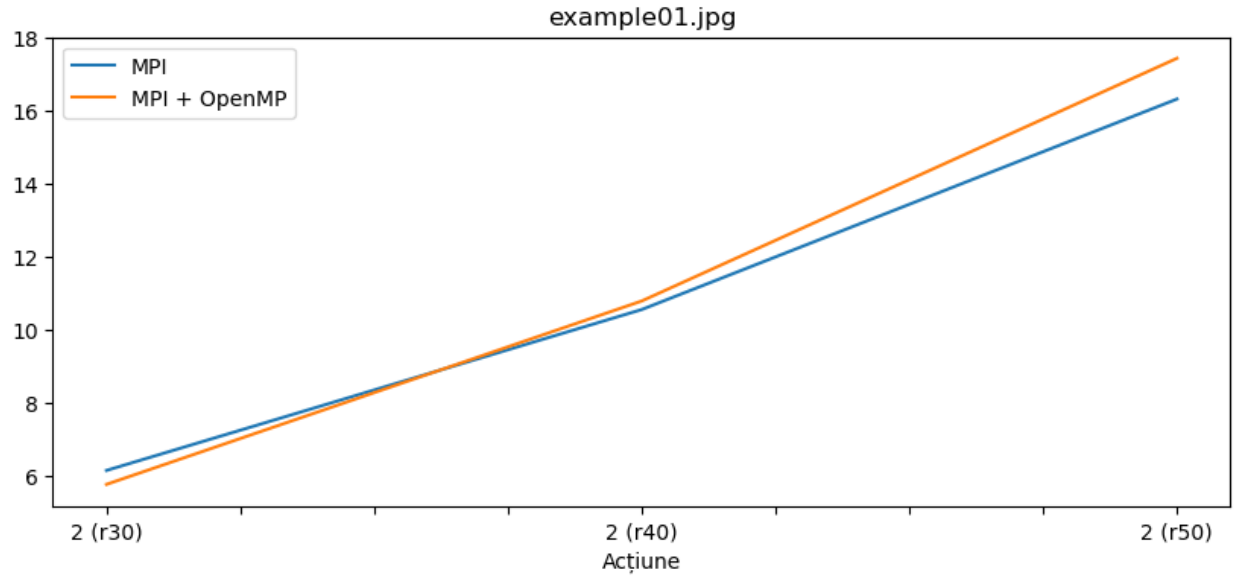

Acțiune #2
MPI
MPI + OpenMP
4
3
2
1
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

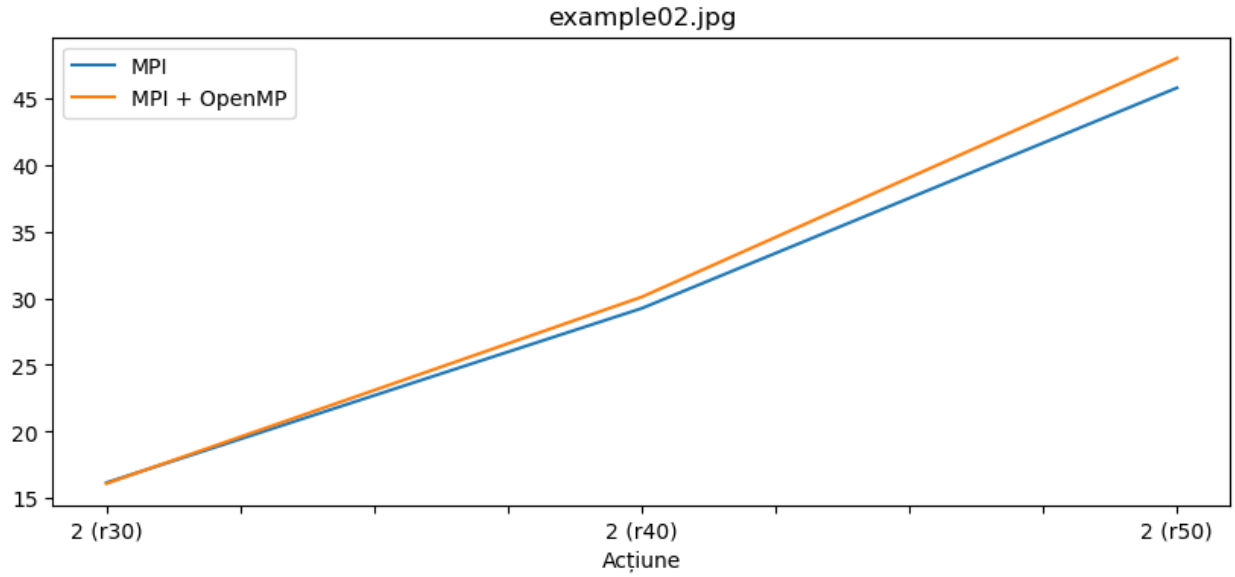

Acțiune #3
MPI
MPI + OpenMP
0.08
0.07
0.06
0.05
0.04
0.03
0.02
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

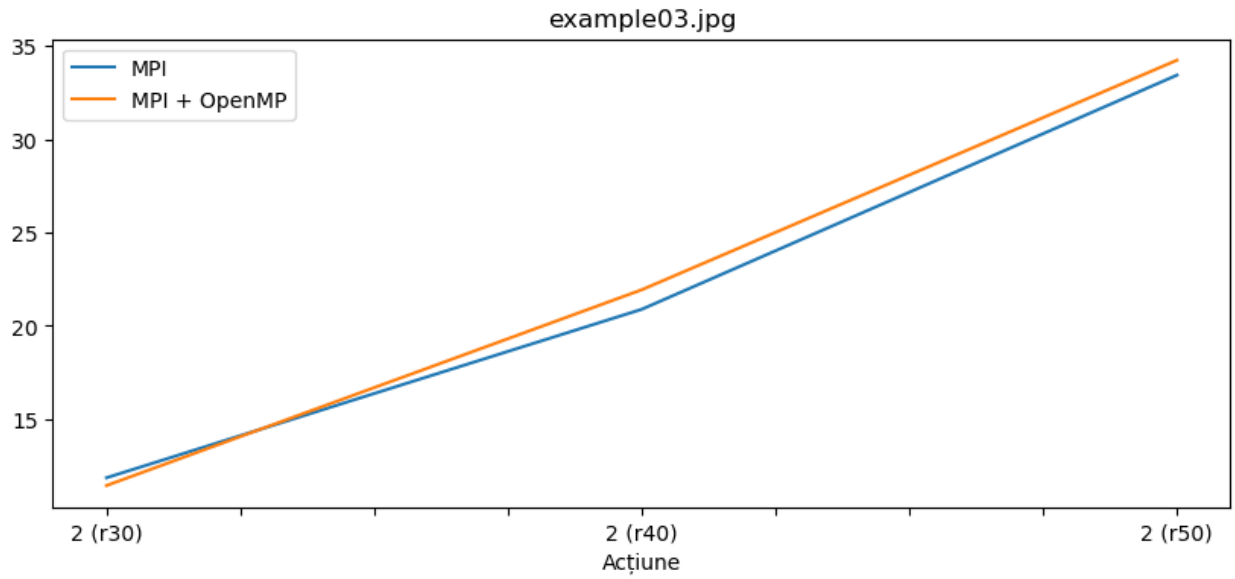
Deja de la 2 procesoare pe WSL începem să avem timpi nesemnificativi pe orice operații în afară de 2 - `ACTION_GAUSSIAN_BLUR`. Creștem numărul de procese la 6 dar vom crește si radiusul kernelului la 30, 40 și 50 pentru a avea diferente semnificative.

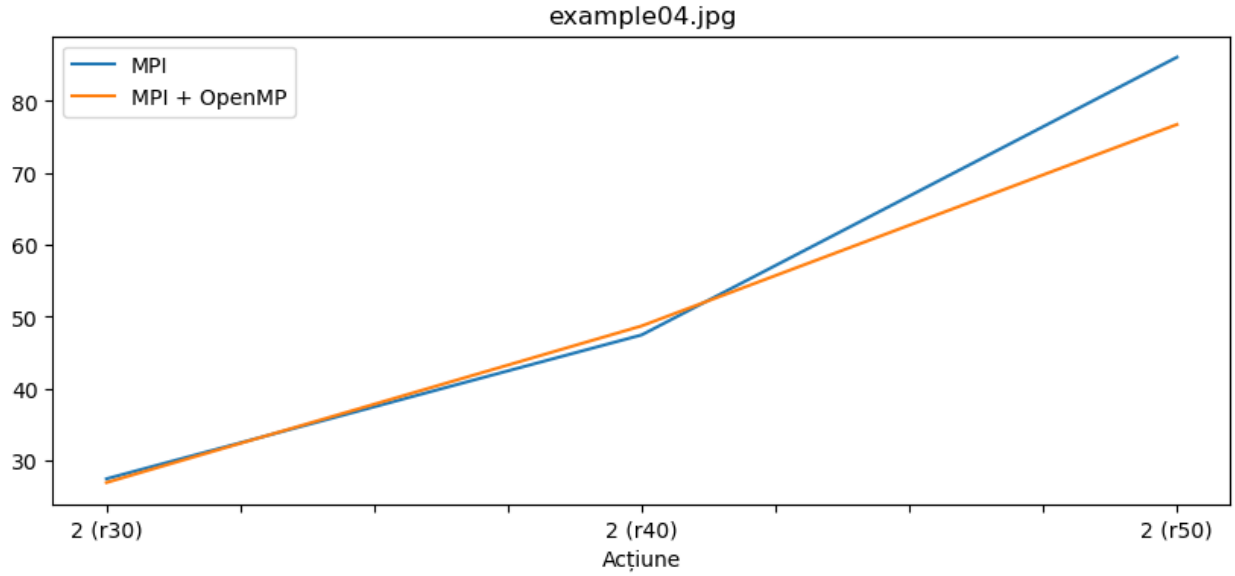
Execuție paralelă (#6) (WSL2) pe aceeași mașină

Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	6	paralelă	2 (r30)	6.154688	5.773731
example01.jpg	3254	4878	6	paralelă	2 (r40)	10.553646	10.787584
example01.jpg	3254	4878	6	paralelă	2 (r50)	16.309375	17.423562
example02.jpg	5504	8256	6	paralelă	2 (r30)	16.157813	16.090529
example02.jpg	5504	8256	6	paralelă	2 (r40)	29.248437	30.103073
example02.jpg	5504	8256	6	paralelă	2 (r50)	45.809375	48.035509
example03.jpg	5494	5830	6	paralelă	2 (r30)	11.839583	11.422477
example03.jpg	5494	5830	6	paralelă	2 (r40)	20.875000	21.922567
example03.jpg	5494	5830	6	paralelă	2 (r50)	33.435938	34.236741
example04.jpg	10315	7040	6	paralelă	2 (r30)	27.435937	26.915831
example04.jpg	10315	7040	6	paralelă	2 (r40)	47.444271	48.696036
example04.jpg	10315	7040	6	paralelă	2 (r50)	86.056771	76.702278


example01.jpg
MPI
MPI + OpenMP
18
16
14
12
10
8
6
2 (r30)
2 (r40)
2 (r50)
Acțiune

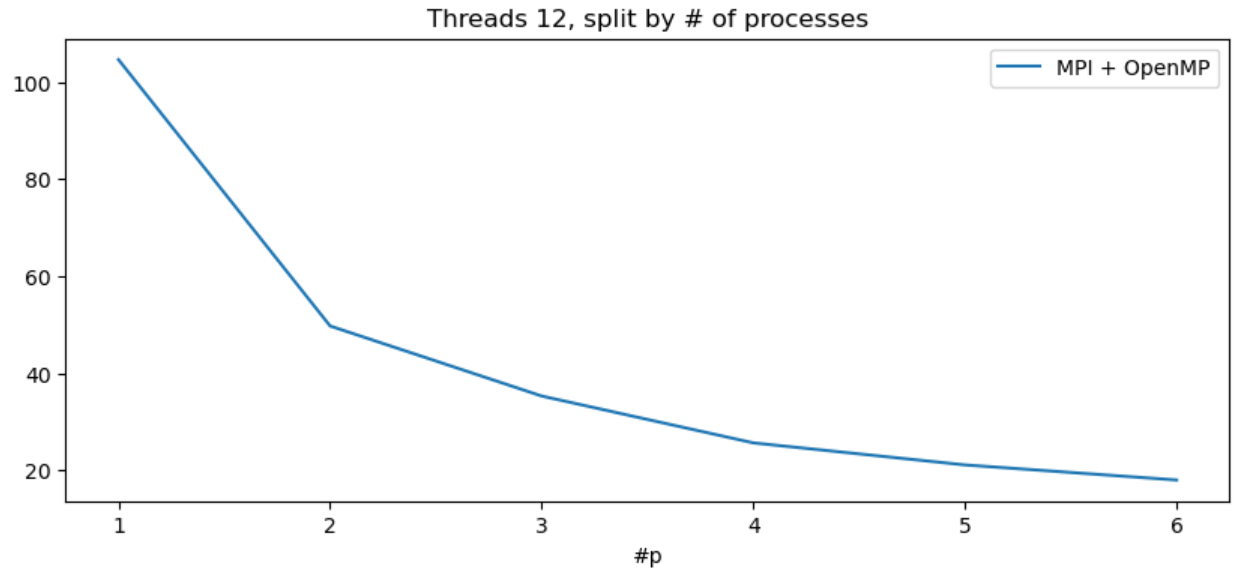

example02.jpg
MPI
MPI + OpenMP
45
40
35
30
25
20
15
2 (r30)
2 (r40)
2 (r50)
Acțiune


example03.jpg
MPI
MPI + OpenMP
35
30
25
20
15
2 (r30)
2 (r40)
2 (r50)
Acțiune


example04.jpg
MPI
MPI + OpenMP
80
70
60
50
40
30
2 (r30)
2 (r40)
2 (r50)
Acțiune

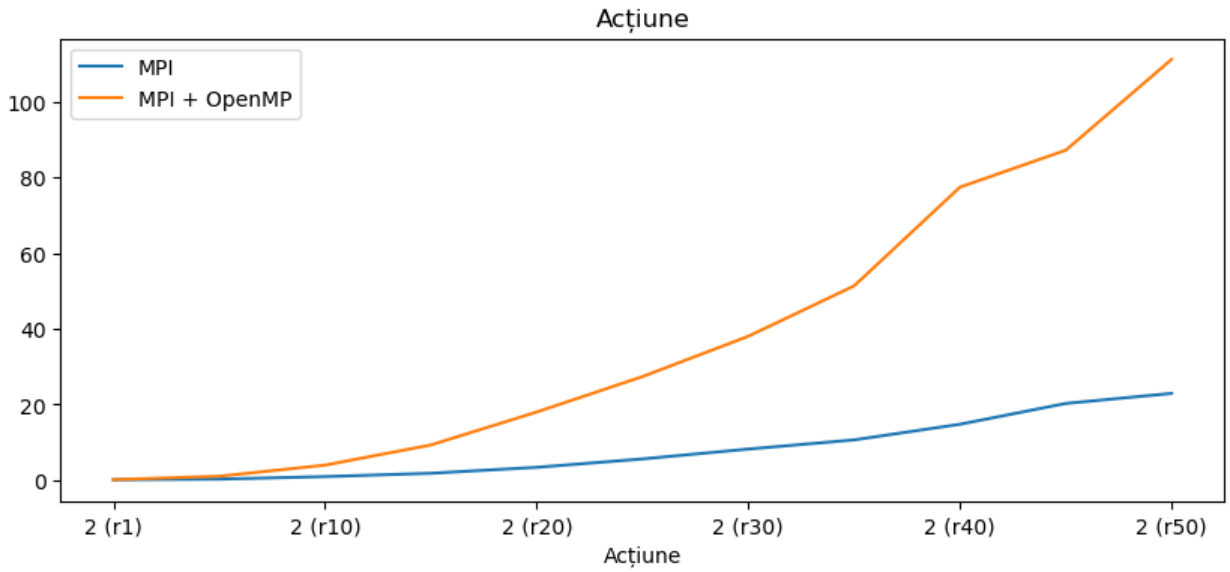
Execuție paralelă (#1-6) (WSL2) pe aceeași mașină

Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI + OpenMP
example01.jpg	3254	4878	1	paralelă	2 (r50)	104.639220
example01.jpg	3254	4878	2	paralelă	2 (r50)	49.760919
example01.jpg	3254	4878	3	paralelă	2 (r50)	35.326322
example01.jpg	3254	4878	4	paralelă	2 (r50)	25.675688
example01.jpg	3254	4878	5	paralelă	2 (r50)	21.135879
example01.jpg	3254	4878	6	paralelă	2 (r50)	18.028109



Execuție secvențială (r1-50) (WSL2) pe aceeași mașină (t12)

Nume	Randuri	Coloane	Tip execuție	Acțiune	MPI (6)	MPI + OpenMP (12)
example01.jpg	3254	4878	secvențială	2 (r1)	0.050052	0.079490
example01.jpg	3254	4878	secvențială	2 (r5)	0.218371	0.979214
example01.jpg	3254	4878	secvențială	2 (r10)	0.884522	3.919819
example01.jpg	3254	4878	secvențială	2 (r15)	1.767596	9.251386
example01.jpg	3254	4878	secvențială	2 (r20)	3.336795	17.965638
example01.jpg	3254	4878	secvențială	2 (r25)	5.556304	27.332810
example01.jpg	3254	4878	secvențială	2 (r30)	8.166545	37.992273
example01.jpg	3254	4878	secvențială	2 (r35)	10.590941	51.319945
example01.jpg	3254	4878	secvențială	2 (r40)	14.727410	77.416420
example01.jpg	3254	4878	secvențială	2 (r45)	20.226182	87.187411
example01.jpg	3254	4878	secvențială	2 (r50)	22.883261	111.197905


Acțiune
MPI
MPI + OpenMP
100
80
60
40
20
0
2 (r1)
2 (r10)
2 (r20)
2 (r30)
2 (r40)
2 (r50)
Acțiune

Cluster VM specificații

Spec	Value
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
Address sizes	39 bits physical, 48 bits virtual
CPU(s)	2
On-line CPU(s) list	0-1
Thread(s) per core	1
Core(s) per socket	1
Socket(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	165
Model name	Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz
Stepping	2
CPU MHz	2711.999

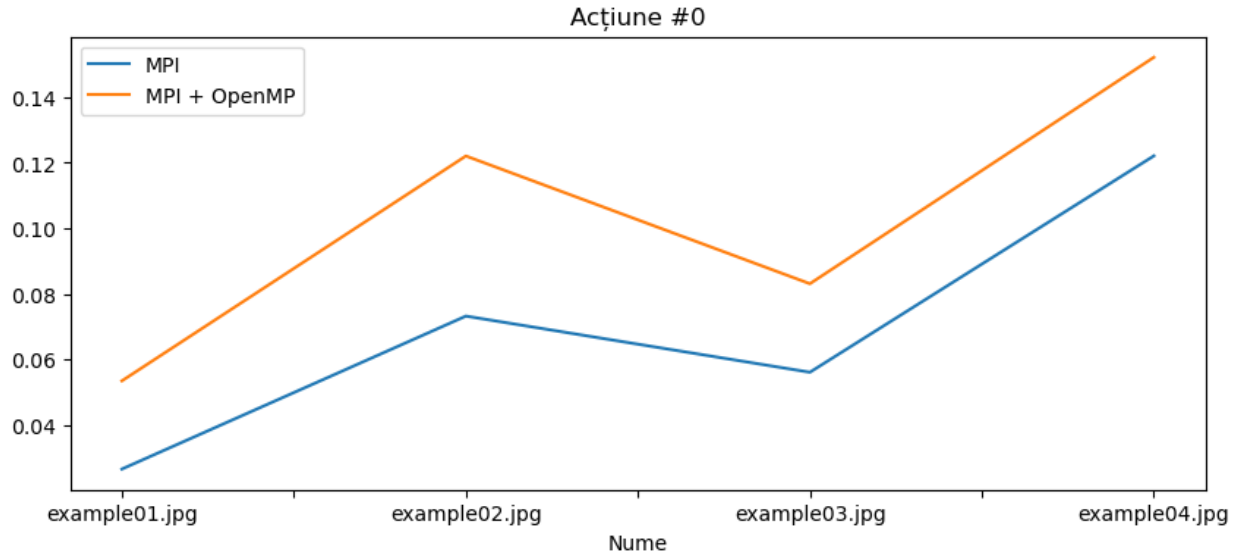
CPU max MHz	2712.000
BogoMIPS	5423.99
Hypervisor vendor	Microsoft
Virtualization type	full
L1d cache	32 KiB
L1i cache	32 KiB
L2 cache	256 KiB
L3 cache	12 MiB
NUMA node0 CPU(s)	0,1
Vulnerability Itlb multihit	KVM: Vulnerable
Vulnerability L1tf	Not affected
Vulnerability Mds	Not affected
Vulnerability Meltdown	Not affected
Vulnerability Spec store bypass	Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1	Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2	Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds	Not affected

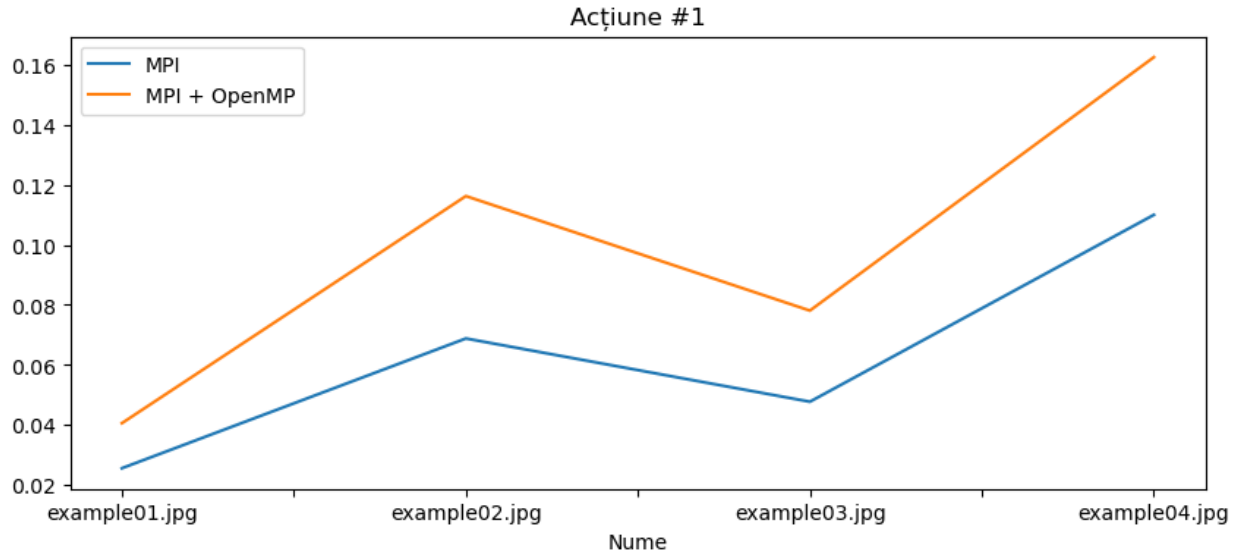
Vulnerability Tsx async abort	Not affected
Flags	fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid pni pclmulqdq ssse3 fma cx16 pcid sse 4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced fsgsb ase bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves flush_l1d arch_capabilities

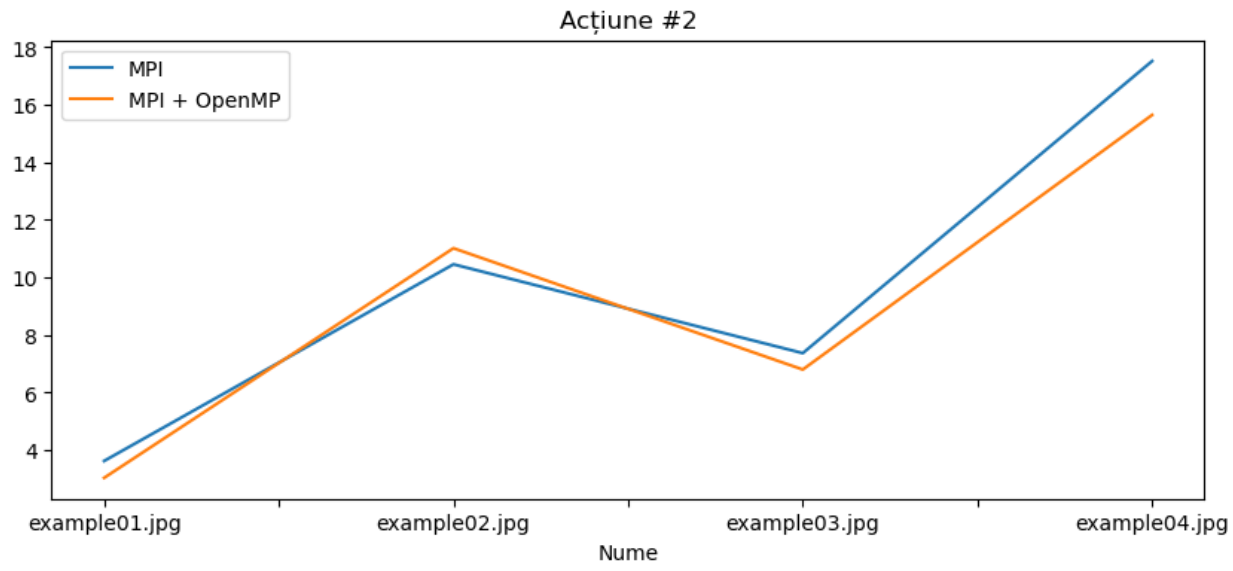
Execuție secvențială (cluster VM)

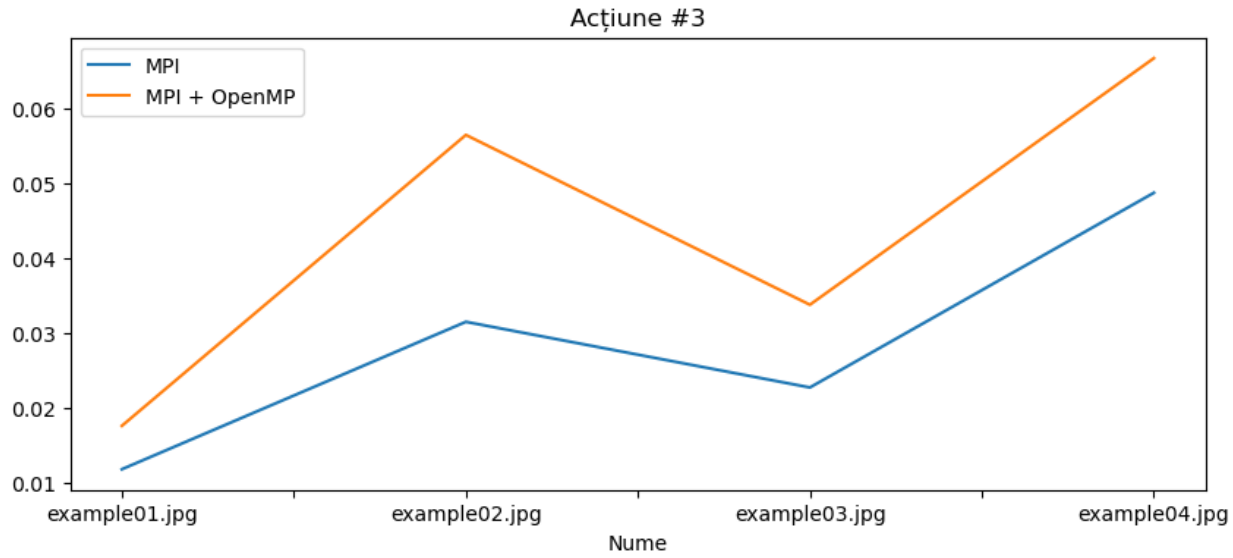
Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	1	secvențială	0	0.026578	0.053484
example01.jpg	3254	4878	1	secvențială	1	0.025662	0.040644
example01.jpg	3254	4878	1	secvențială	2 (r5)	3.622666	3.034894
example01.jpg	3254	4878	1	secvențială	3	0.011744	0.017546
example02.jpg	5504	8256	1	secvențială	0	0.073243	0.122090
example02.jpg	5504	8256	1	secvențială	1	0.068831	0.116207
example02.jpg	5504	8256	1	secvențială	2 (r5)	10.456979	11.016649
example02.jpg	5504	8256	1	secvențială	3	0.031446	0.056425
example03.jpg	5494	5830	1	secvențială	0	0.056116	0.083122
example03.jpg	5494	5830	1	secvențială	1	0.047791	0.078103
example03.jpg	5494	5830	1	secvențială	2 (r5)	7.369922	6.800011
example03.jpg	5494	5830	1	secvențială	3	0.022672	0.033729
example04.jpg	10315	7040	1	secvențială	0	0.122126	0.152197
example04.jpg	10315	7040	1	secvențială	1	0.109964	0.162415
example04.jpg	10315	7040	1	secvențială	2 (r5)	17.521252	15.648024

example04.jpg	10315	7040	1	secvențială	3	0.048693	0.066684





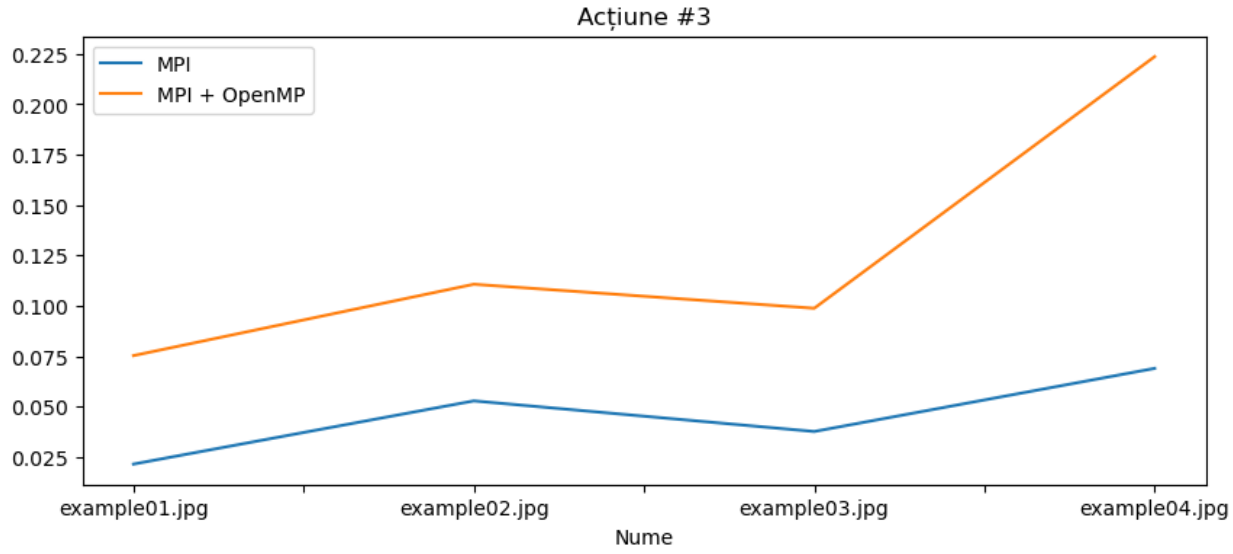

Acțiune #2
MPI
MPI + OpenMP
18
16
14
12
10
8
6
4
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

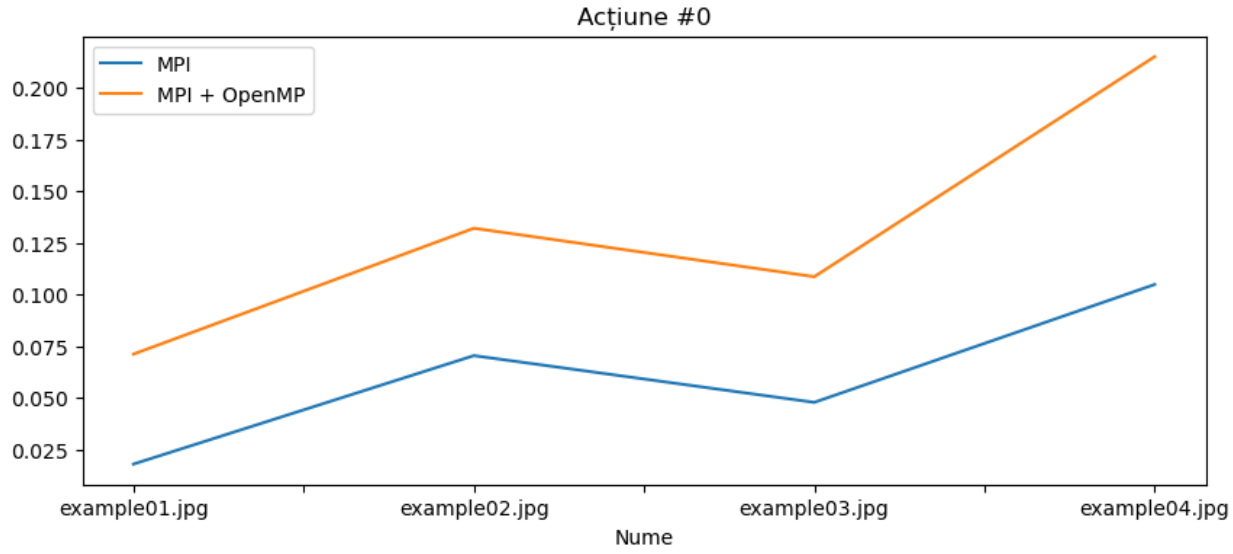

Acțiune #3
MPI
MPI + OpenMP
0.06
0.05
0.04
0.03
0.02
0.01
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

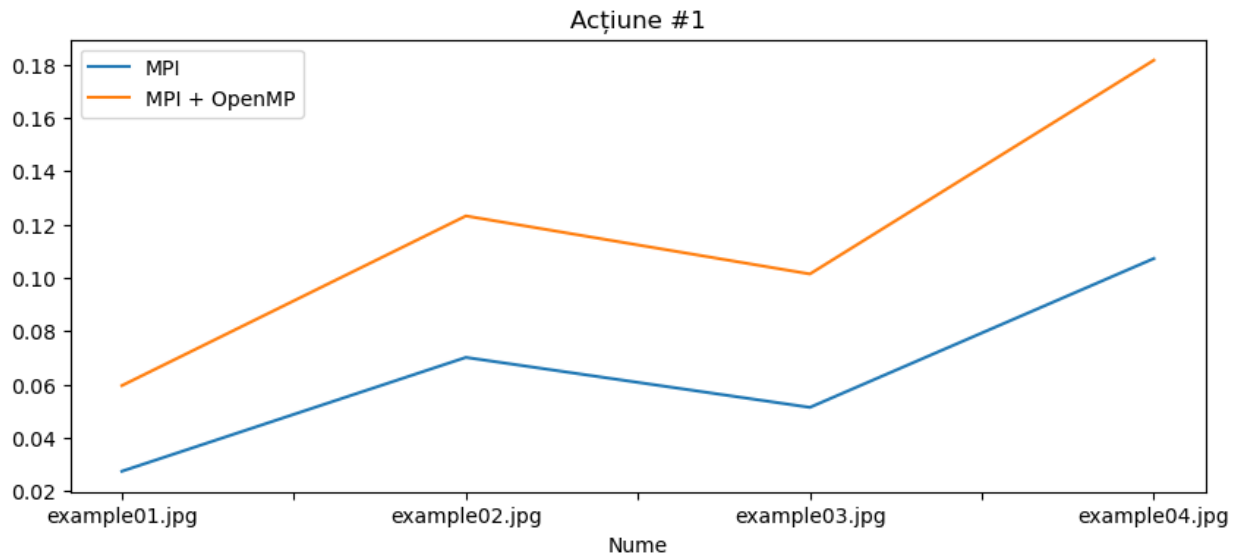
Execuție paralelă (#2) (cluster VM) pe aceeași mașină

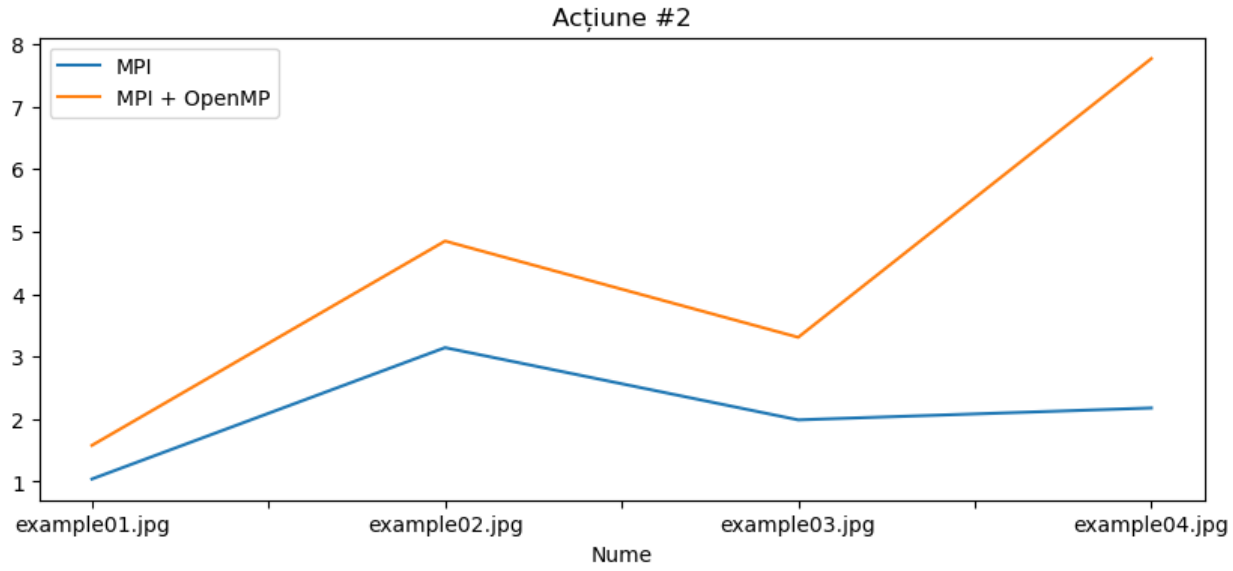
Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	2	paralelă	0	0.017986	0.071200
example01.jpg	3254	4878	2	paralelă	1	0.027423	0.059566
example01.jpg	3254	4878	2	paralelă	2 (r5)	1.041350	1.579876
example01.jpg	3254	4878	2	paralelă	3	0.021421	0.075241
example02.jpg	5504	8256	2	paralelă	0	0.070463	0.132092
example02.jpg	5504	8256	2	paralelă	1	0.070134	0.123229
example02.jpg	5504	8256	2	paralelă	2 (r5)	3.143462	4.849303
example02.jpg	5504	8256	2	paralelă	3	0.052785	0.110653
example03.jpg	5494	5830	2	paralelă	0	0.047854	0.108646
example03.jpg	5494	5830	2	paralelă	1	0.051390	0.101450
example03.jpg	5494	5830	2	paralelă	2 (r5)	1.987500	3.309027
example03.jpg	5494	5830	2	paralelă	3	0.037593	0.098739
example04.jpg	10315	7040	2	paralelă	0	0.104874	0.215122
example04.jpg	10315	7040	2	paralelă	1	0.107259	0.181679
example04.jpg	10315	7040	2	paralelă	2 (r5)	2.177675	7.768345

example04.jpg	10315	7040	2	paralelă	3	0.068907	0.223593





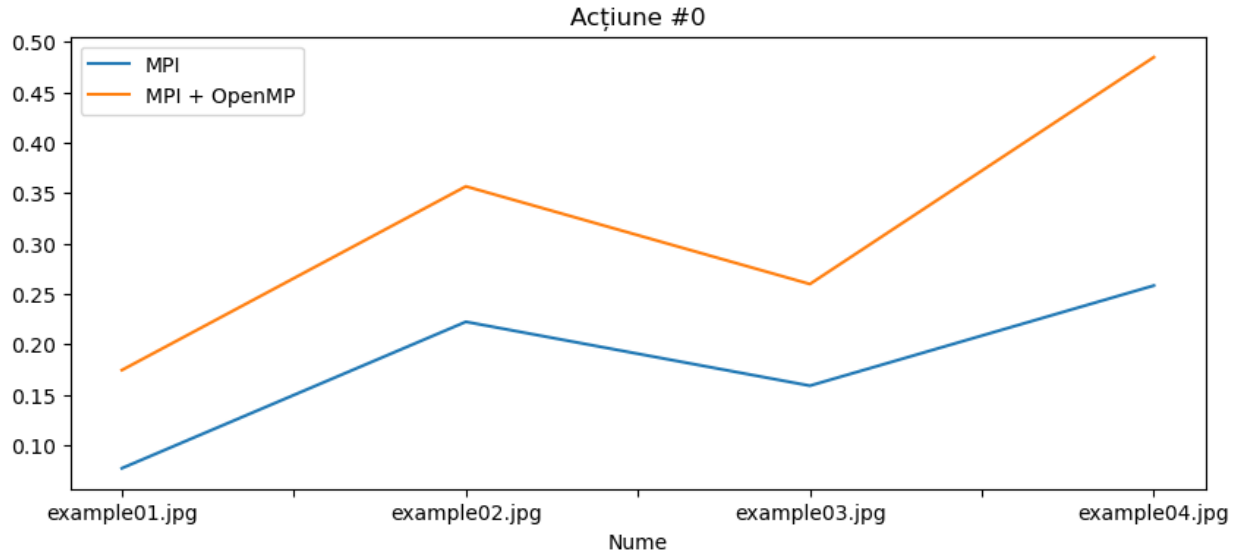

Acțiune #1
MPI
MPI + OpenMP
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

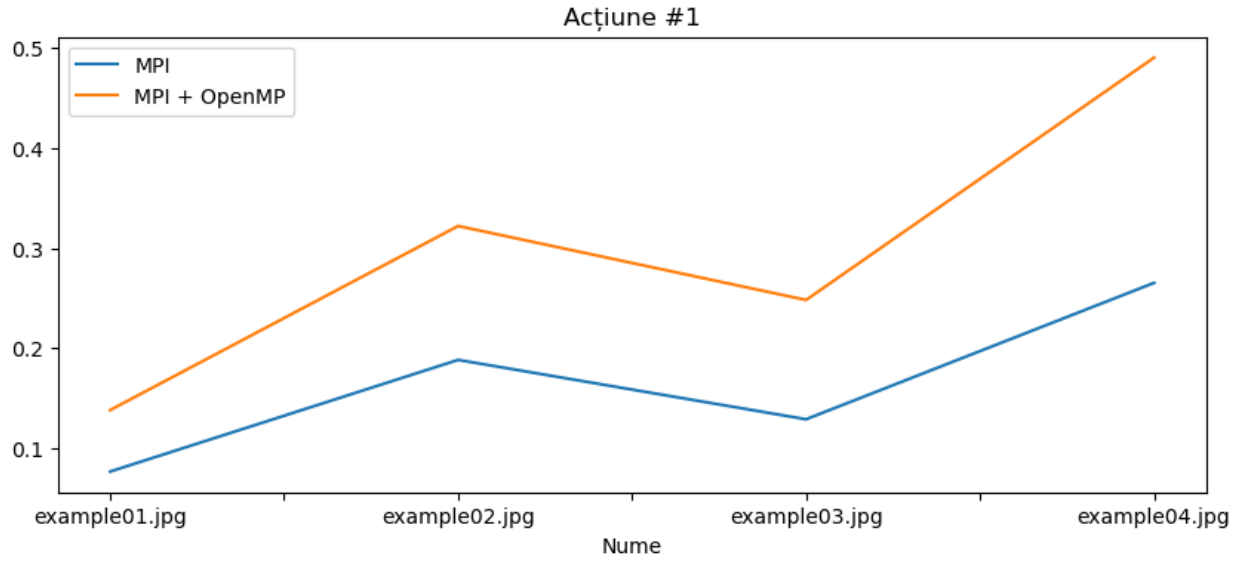

Acțiune #2
MPI
MPI + OpenMP
8
7
6
5
4
3
2
1
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

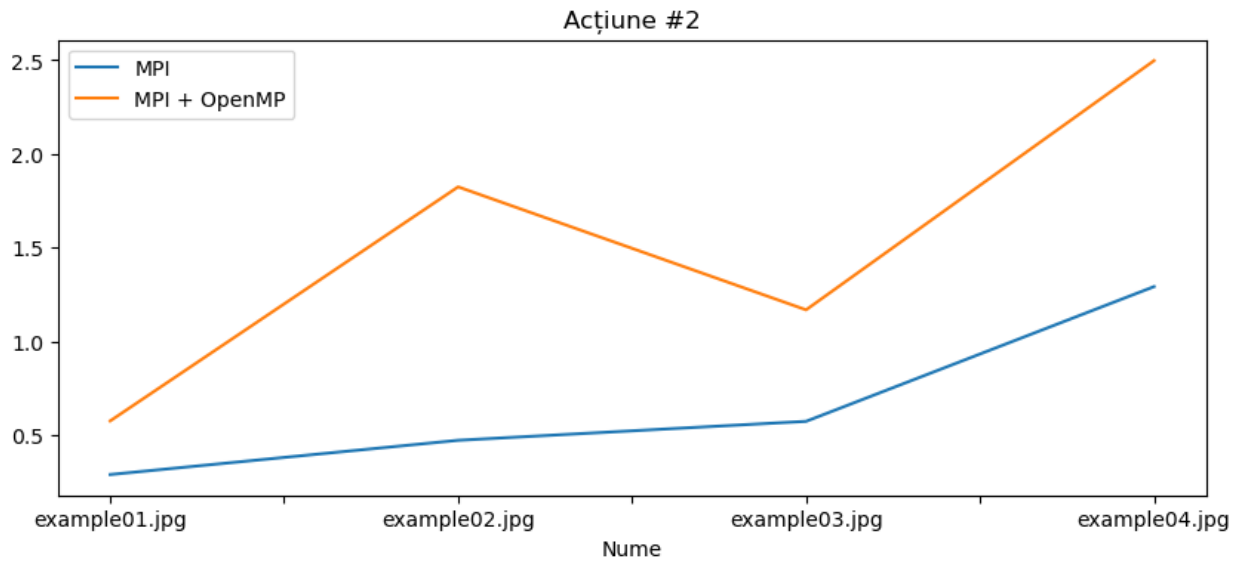
Execuție paralelă (#6) (cluster VM) pe 3 mașini

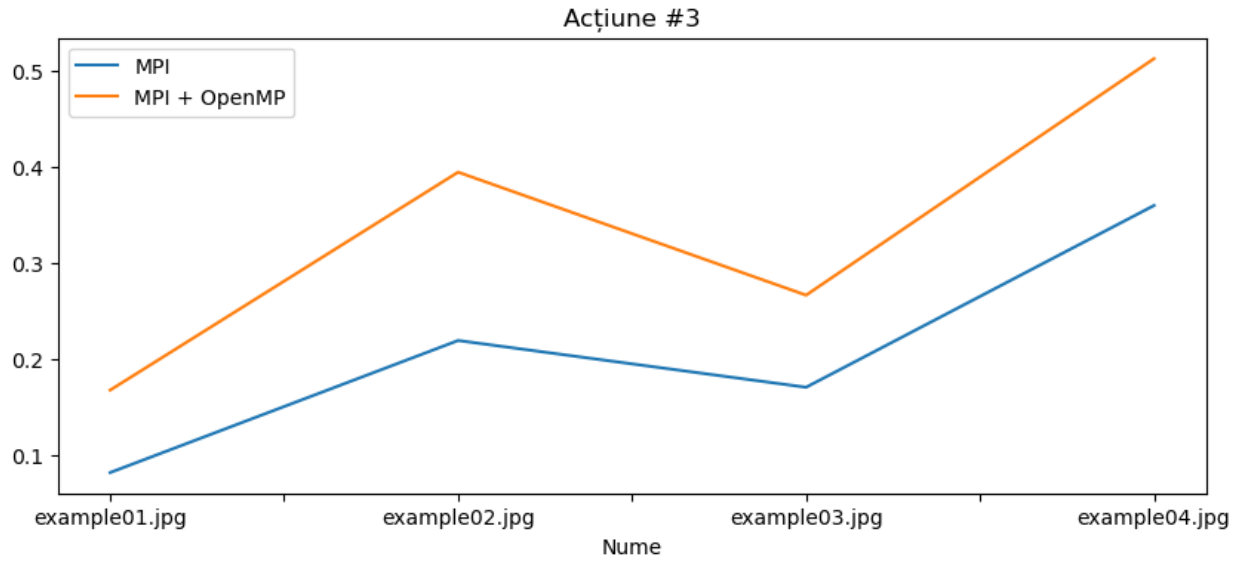
Nume	Randuri	Coloane	#p	Tip execuție	Acțiune	MPI	MPI + OpenMP
example01.jpg	3254	4878	6	paralelă	0	0.076858	0.174233
example01.jpg	3254	4878	6	paralelă	1	0.076863	0.138247
example01.jpg	3254	4878	6	paralelă	2 (r5)	0.289518	0.575545
example01.jpg	3254	4878	6	paralelă	3	0.082530	0.168109
example02.jpg	5504	8256	6	paralelă	0	0.222178	0.356577
example02.jpg	5504	8256	6	paralelă	1	0.188356	0.322068
example02.jpg	5504	8256	6	paralelă	2 (r5)	0.472114	1.824033
example02.jpg	5504	8256	6	paralelă	3	0.219584	0.394362
example03.jpg	5494	5830	6	paralelă	0	0.158759	0.259641
example03.jpg	5494	5830	6	paralelă	1	0.129069	0.248196
example03.jpg	5494	5830	6	paralelă	2 (r5)	0.573328	1.168061
example03.jpg	5494	5830	6	paralelă	3	0.171056	0.266668
example04.jpg	10315	7040	6	paralelă	0	0.258141	0.484788
example04.jpg	10315	7040	6	paralelă	1	0.265265	0.490213
example04.jpg	10315	7040	6	paralelă	2 (r5)	1.291921	2.497428

example04.jpg	10315	7040	6	paralelă	3	0.359761	0.512207

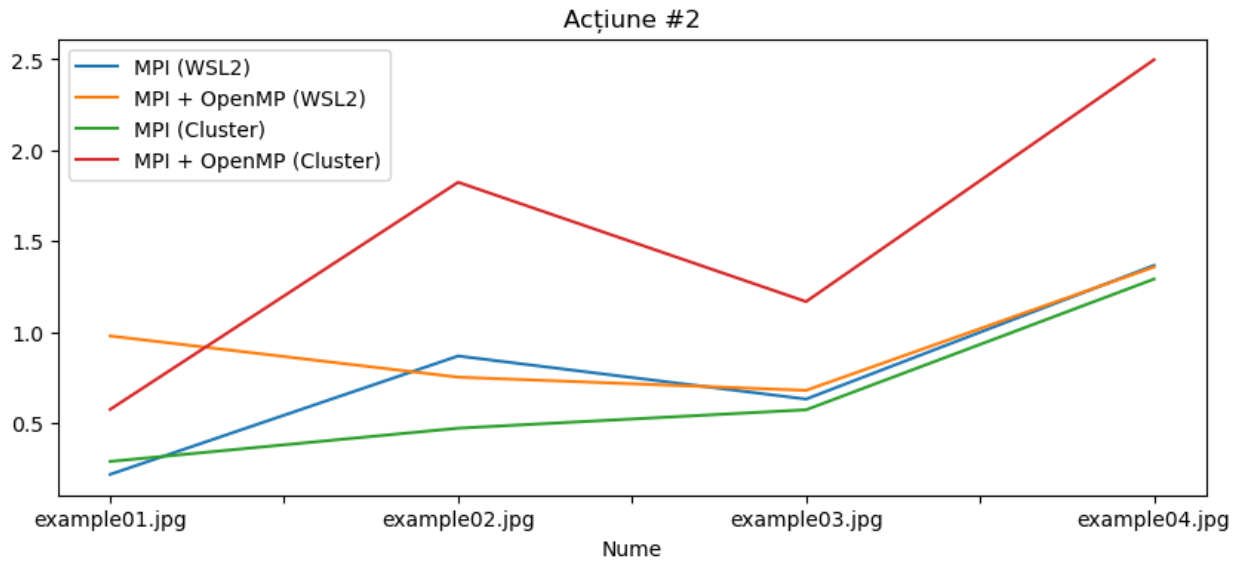





Acțiune #2
MPI
MPI + OpenMP
2.5
2.0
1.5
1.0
0.5
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume


Acțiune #3
MPI
MPI + OpenMP
0.5
0.4
0.3
0.2
0.1
example01.jpg
example02.jpg
example03.jpg
example04.jpg
Nume

Execuție paralelă (#6) (WSL2) pe aceeași mașină vs Execuție paralelă (#6) (cluster VM) pe 3 mașini



CONCLUZIE

Din Tema 1:

Execuția secvențială în WSL2 este mai înceată decât pe un VM full folosind Hyper-V.
Execuția paralelizată pe WSL2 (o singură mașină) este mai rapidă decât execuția pe 3 VMuri full folosind Hyper-V.
Costul este simțitor atunci când datele de input sunt multe ca număr (date fiind întârzierile provocate de transferul datelor în rețea (ssh)) deși taskurile sunt procesate mai rapid pe VMuri.
Când taskurile sunt mai mari (matricea sau inputul unui singur file este mult mai mare) atunci merită plătit costul rețelei pentru transferul datelor.
Diferențele duratei de procesare între imaginile care au număr de pixeli diferiți sunt date, în principal, de numărul de rânduri (atât scatter/gather cât și send/receive procesează sliceuri de rânduri).
Împărțirea prin scatter/gather sau send/receive este optimă pentru fiecare caz - dacă dimensiunile matricilor cresc sau radiusul blurului (o altă matrice) se mărește, creșterea duratei de procesare nu este exponențială.

Apoi, punând OpenMP peste MPI în Tema 3:

Execuțiile acțiunilor exceptând acțiunea 3 sunt mai rapide (parallel (collapsed) for) în unele cazuri dar pe dimensiunile matricilor folosite câștigul nu este enorm/exponențial (de așteptat). Sunt și cazuri în care overheadul generat de managementul threadurilor este mai mare (dar considerăm timpi de ordinul zecilor de milisecunde).

Acțiunea 2 (Gaussian Blur) a ridicat multiple probleme aplicând OpenMP peste. Overheadul creat de OpenMP peste bufferul (matricea) folosită este simțitor și nerecomandabil indiferent de tipul acțiunilor pretabile din Tema 2 folosite. Există un locking făcut pe bufferul de output și s-a resimțit pe cea mai mare matrice (image04) fiind adus la peste 100 de secunde de la 16 (experimental am încercat câteva variabile volatile în scope și a dispărut overheadul). Diferența enormă este dată de faptul că în împărțirea făcută pe MPI nu există o zonă shared, toate submatricile sunt copiate și trimise pe când în zona OpenMPului toate submatricile sunt shared între numărul de threaduri. Ținând

cont de aceste observații, concluzia este că cea mai bună optimizare pentru acțiunea 2 este lock-free - multiprocessing - și a nu folosi deloc multithreading.