Εργασία Παράλληλων Συστημάτων 2019-20

README Τεκμηρίωση, η Παρουσίαση και τα **Αποτελέσματα των προγραμμάτων**(Το README για τον κώδικα βρίσκεται στο άλλο README_CODE)

Game of Life

Μέλη ομάδας:
Φαραώ Γεώργιος 1115201700177

Στρόμπολας Σοφοκλής 1115201700153

1. Εισαγωγή

Αυτή η εργασία σχεδιάζει, υλοποιεί και αξιολογεί το **Conway's Game of Life** σε περιβάλλοντα MPI και Υβριδικό MPI+OpenMp.

Ο κώδικας βρίσκεται στα αρχεία **gol_mpi.c** και **gol_mpi_omp.c** και επεξηγείται περισσότερο στο αρχείο README_CODE και η εκτέλεση για όλες τις μετρήσεις έγινε στο μηχάνημα ΑΡΓΩ.

Εχουμε χρησιμοποιήσει 500 generations για όλες τις μετρήσεις, ώστε ο αριθμός επαναλήψεων να είναι σταθερός, όπως αναφέρεται στην εκφώνηση της άσκησης. Όσον αφορά τα grid δηλαδή τα μεγέθη του πίνακα , χρησιμοποιήσαμε αριθμούς που να διαιρούνται ακριβώς από την ρίζα των αριθμών των διεργασιών ώστε το καρτεσιανό που βγαίνει να είναι τετραγωνικό. Για αυτό για διεργασίες 1,4,9,16,25,64 χρησιμοποιούμε τους πίνακες 120x120, 240x240,480x480,960x960,3840x3840, και σε συγκεκριμένη περίπτωση για κάποιες μετρήσεις δοκιμάσαμε και 15360x15360. Για όλες τις περιπτώσεις έχει γίνει διάγραμμα και μελέτη με σχολιασμό των αποτελεσμάτων.

2. Σχεδιασμός Διαμοιρασμού Δεδομένων

Το πως υλοποιείται ο σχεδιασμός το έχουμε αναφέρει και στο README_CODE, αλλά και στα σχόλια των προγραμμάτων. Σε γενικά πλαίσια όμως ο σχεδιασμός διαμοιρασμού δεδομένων έχει γίνει ως εξής:

Πρώτα φτιάχνεται το αρχικό grid που είναι φτιαγμένο ώς τόρος. Υστερα δημιουργείται και το block για κάθε διεργασία..

Κάνοντας scatter μοιράζουμε το αρχικό μας grid στα processes αναλογα με τις αρχικές συντεταγμένες του καθενός.

΄Υστερα βρίσκουμε τα γειτονικά blocks ώστε να ξέρουμε από ποιους θα πάρουμε και σε ποιους θα δώσουμε. Κάθε block έχει 8 γειτονικές επαφές, αφού το grid μας είναι τόρος έχει περιοδικότητα.

Και μετά τρέχουμε τα generations , και γίνονται οι κατάλληλες ανταλλαγές μεταξύ των διεργασιών ώστε να βρεθεί το αποτέλεσμα.

Κάνοντας gather μαζεύουμε όλα τα τελικά blocks στο αρχικό μας grid και έτσι βρίσκουμε το τελικό μας grid.

3. Σχεδιασμός MPI κώδικα - Βελτιστοποιήσεις

Η υλοποίηση του MPI κώδικα έχει σχολιαστεί μέσα στα αρχεία κώδικα, αλλά και στο README_CODE, οπότε σε αυτό το κομμάτι αναφέρουμε τις βελτιστοποιήσεις που κάναμε, ύστερα από δοκιμές διαφόρων μεθόδων και παροτρύνσεις της εκφώνησης.

- Διαμοιράζουμε τα δεδομένα σε block, όπου κάθε διεργασία αναλαμβάνει και από ένα block.
- Χρησιμοποιούμε ένα βοηθητικό πίνακα που αποθηκεύουμε τις αλό και τα γωνιακά σημεία. Αντι δηλαδή να φτιάχνουμε έναν νέο N+2 x M+2 πίνακα κάθε φορά για κάθε μπλοκ, απλώς βάλαμε το μπλοκ να κρατάει σε εναν 8 x blockside array τα περιεχόμενα που μας γίνονται receive.
- Κάνουμε πρώτα τις εντολές Receive και μετά τις Send.
- Στέλνουμε ολόκληρες γραμμές και στήλες μαζί, όχι ως μεμονωμένα σημεία.
- Στα γωνιακά αλό σημεία που πρέπει να στείλουμε, στέλνεται μόνο η γωνία και όχι όλη η στήλη.
- Υπολογίζουμε μία φορά, έξω από την κεντρική επανάληψη, τα rank των γειτόνων.
- Οι διεργασίες που επικοινωνούν βρίσκονται στον ίδιο κόμβο.
- Χρησιμοποιούμε δυναμικούς πίνακες και δεν καλούμε συναρτήσεις μέσα στην κεντρική επανάληψη.
- Κάθε 10 επαναλήψεις γίνεται έλεγχος για μη αλλαγή ή μηδενισμό του πλέγματος.
- Οι τιμές για το reduce γίνεται κατά τον υπολογισμό των νέων τιμών και όχι με νέα for.
- Για το OpenMP παραλληλοποιούμε την διπλή for για τα εσωτερικά σημεία και τις 4 for για τα εξωτερικά.
- Τρέξαμε το πρόγραμμα και βρήκαμε τους καλύτερους συνδυασμούς διεργασιών και νημάτων.

4. Μετρήσεις και Μελέτη

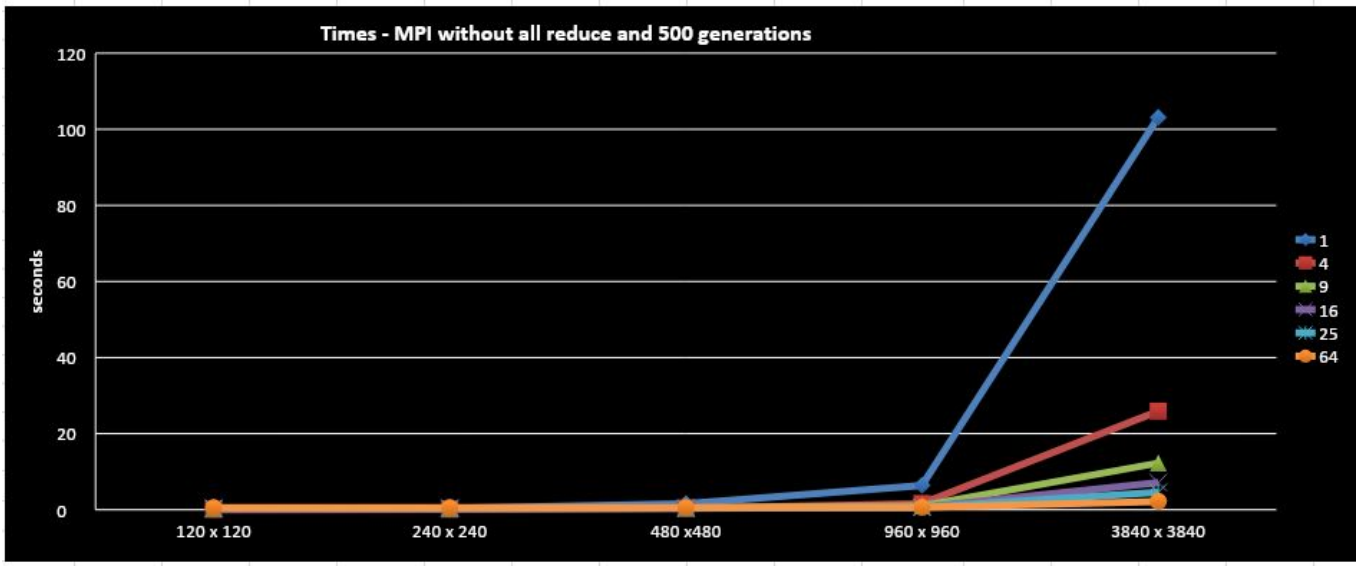
Οι μετρήσεις πραγματοποιήθηκαν στο μηχάνημα Αργώ.

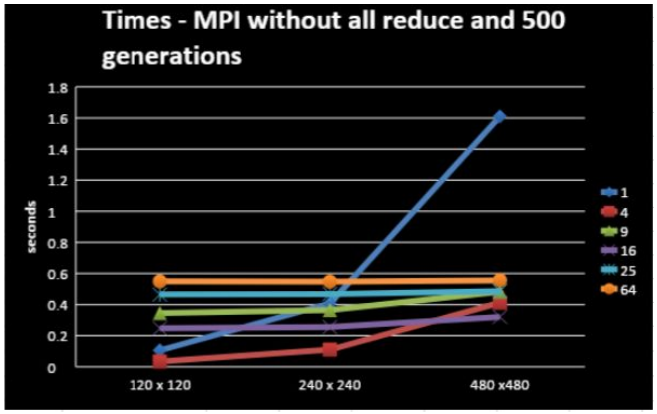
a. MPI πρόγραμμα, χωρίς allreduce

i. Χρόνοι

	MPI without all reduce and 500 generations					
grid\processes	1	4	9	16	25	64
120 x 120	0,105703	0,033863	0,345056	0,247007	0,464553	0,549173
240 x 240	0,406705	0,110106	0,365109	0,255157	0,46816	0,547408
480 x480	1,609970	0,412082	0,482973	0,321280	0,487076	0,555872
960 x 960	6,428681	1,624519	1,000759	0,622481	0,672283	0,624696
3840 x 3840	103,138782	25,889256	12,271297	7,139757	4,593337	2,199084

Το 1ο διάγραμμα απεικονίζει τους χρόνους εκτέλεσης του MPI προγράμματος without allreduce για 500 generations, για όλα τα μεγέθη πινάκων που έγινε έλεγχος, και για τις αντίστοιχες διεργασίες. Στο 2ο παρουσιάζονται μόνο τα μικρά μεγέθη για να φανούν καλύτερα οι χρόνοι.





Παρατηρούμε ότι για μικρά grids (120x120, 240x240,480x480), δεν παρατηρείται σημαντική βελτίωση στον χρόνο εκτέλεσης. Και σε αυτά τα μικρά grids, ο χρόνος εκτέλεσης για λιγότερα processes όπως 4 ή 16 , μπορεί να είναι μικρότερος από τον χρόνο εκτέλεσης για περισσότερα.
Αυτό μπορεί να οφείλεται στο ότι στα πολλα processes, ο χρόνος που δαπανάται στην επικοινωνία μεταξύ των πολλών διεργασιών είναι μεγαλύτερος από το χρόνο που κερδίζεται από τους παράλληλους υπολογισμούς.

Αντίθετα στα μεγάλα grids παρατηρείται σημαντική μείωση του χρόνου εκτέλεσης όσο αυξάνεται ο αριθμός των διεργασιών, όπως και αναμέναμε.

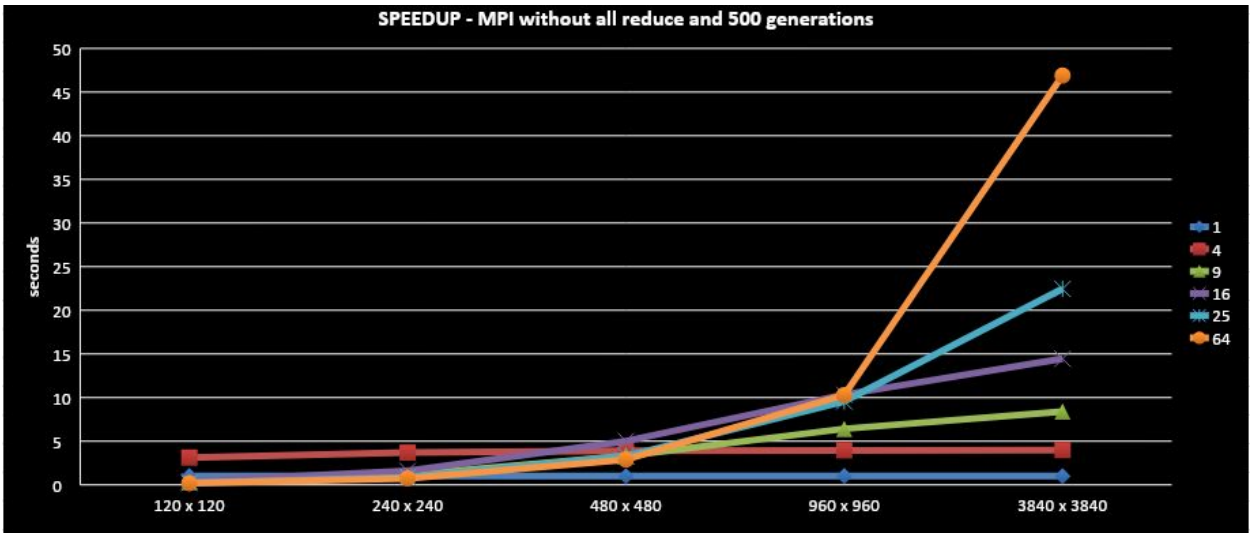
Επίσης από τα 2 διαγράμματα ξεχωρίζει η σημαντική αύξηση του χρόνου εκτέλεσης για 1 διεργασία όσο αυξάνεται το μέγεθος του πίνακα.

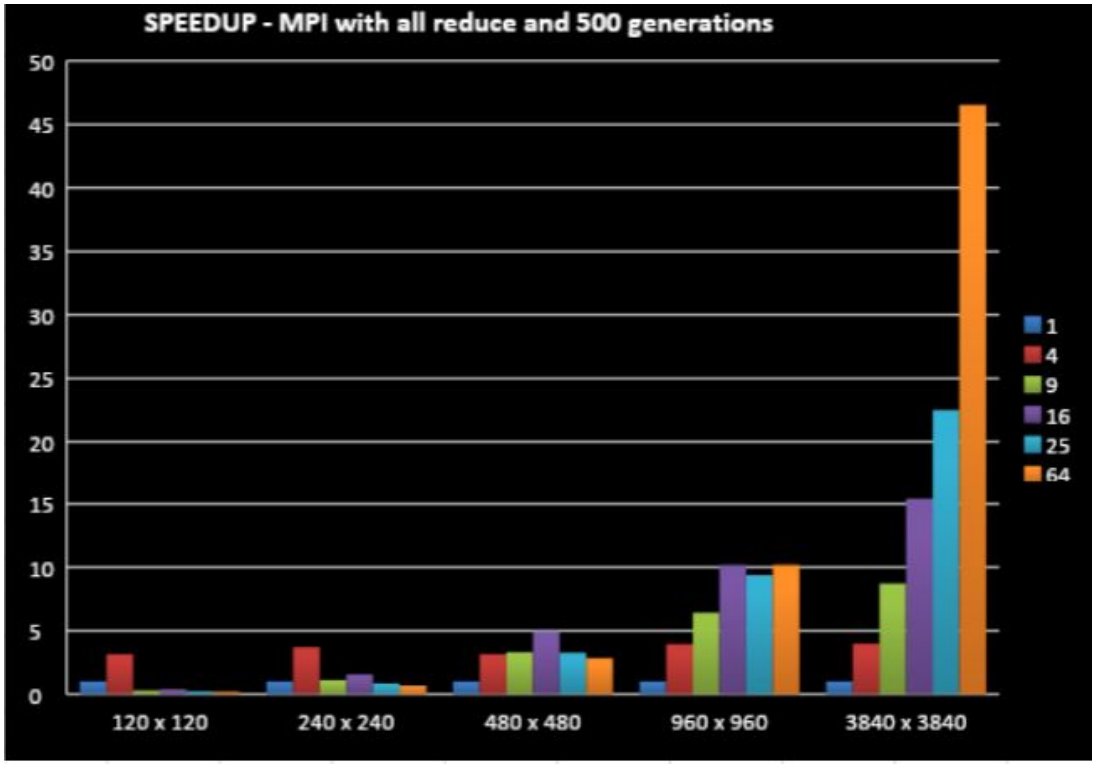
ii.Επιτάχυνση

Επιτάχυνση(Speedup) είναι ο χρόνος του σειριακού προγράμματος προς το χρόνο του κάθε παράλληλου. Δηλαδή S = Ts / Tp.

Επιτάχυνση S(side,p) = Ts(side) / T(side,p)						
	MPI without all reduce and 500 generations					
grid\processes	1	4	9	16	25	64
120 x 120	1	3,121489531	0,3063357832	0,4279352407	0,2275370087	0,1924766877
240 x 240	1	3,693758742	1,113927622	1,593940202	0,8687307758	0,7429650279
480 x 480	1	3,906916585	3,333457564	5,011111803	3,305377395	2,89629627
960 x 960	1	3,95728274	6,423805332	10,32751361	9,562462534	10,29089509
3840 x 3840	1	3,983844959	8,404880267	14,4456992	22,45399848	46,90079233

Τα διαγράμματα παρουσιάζουν την Επιτάχυνση χωρίς allreduce για 500 generations



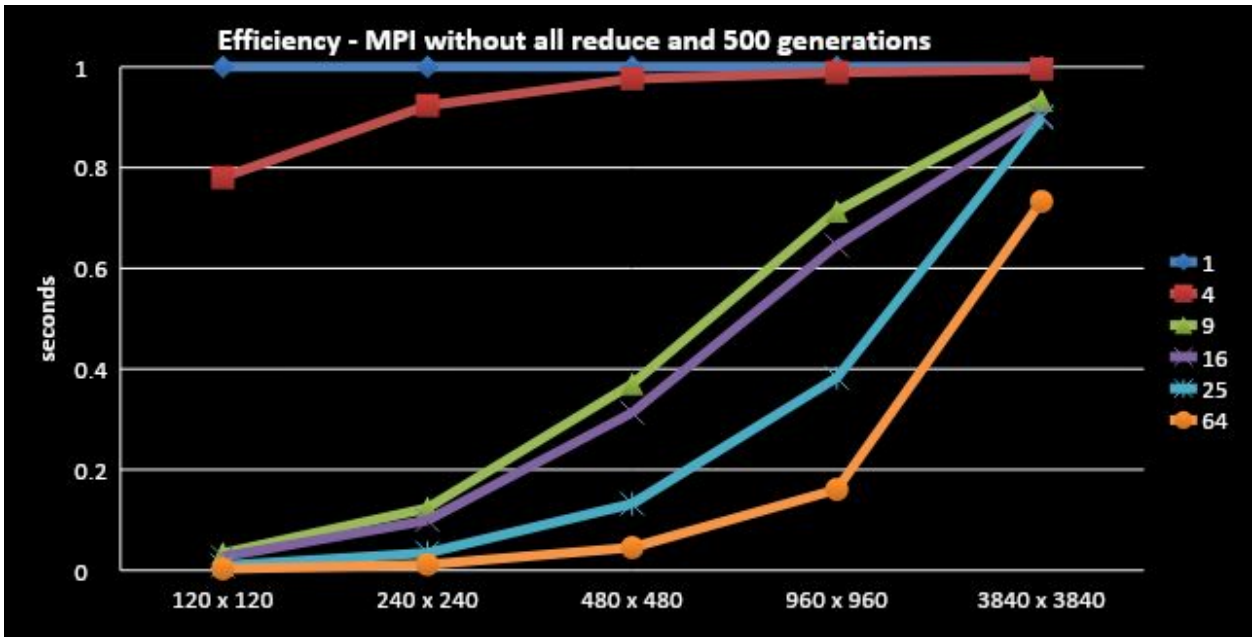


Παρατηρείται σημαντική αύξηση στην επιτάχυνση καθώς αυξάνεται το μέγεθος του grid και ο αριθμός των διεργασιών.Δηλαδή στα μικρά grids, οι λιγότερες διεργασίες έχουν μεγάλη επιτάχυνση, ενώ στα μεγαλύτερα grids οι περισσότερες διεργασίες έχουν μεγάλη επιτάχυνση. Άρα στα μεγαλύτερα grids, το πρόγραμμα συμπεριφέρεται καλύτερα με πολλές διεργασίες.

iii. Αποδοτικότητα

Αποδοτικότητα είναι η Επιτάχυνση που παρουσιάζεται από πάνω , προς τον αριθμό των διεργασιών.E(side, p)=S(side, p)/p.

Αποδοτικότητα		MPI without all reduce and 500 generations					
E(side,p)=S(side,p)/p	grid\processes	1	4	9	16	25	64
	120 x 120	1	0,7803723828	0,03403730924	0,02674595254	0,009101480348	0,003007448245
	240 x 240	1	0,9234396854	0,1237697357	0,0996212626	0,03474923103	0,01160882856
	480 x 480	1	0,9767291461	0,3703841738	0,3131944877	0,1322150958	0,04525462921
	960 x 960	1	0,9893206851	0,713756148	0,6454696007	0,3824985014	0,1607952358
	3840 x 3840	1	0,9959612397	0,9338755852	0,9028562001	0,8981599391	0,7328248802



Παρατηρείται ότι η αποδοτικότητα με μικρότερο αριθμό διεργασιών, για μεγέθη που τρέξαμε είναι καλύτερη από τις περισσότερες διεργασίες. Δηλαδή η αποδοτικότητα όσο αυξάνουμε τις διεργασίες, μειώνεται.

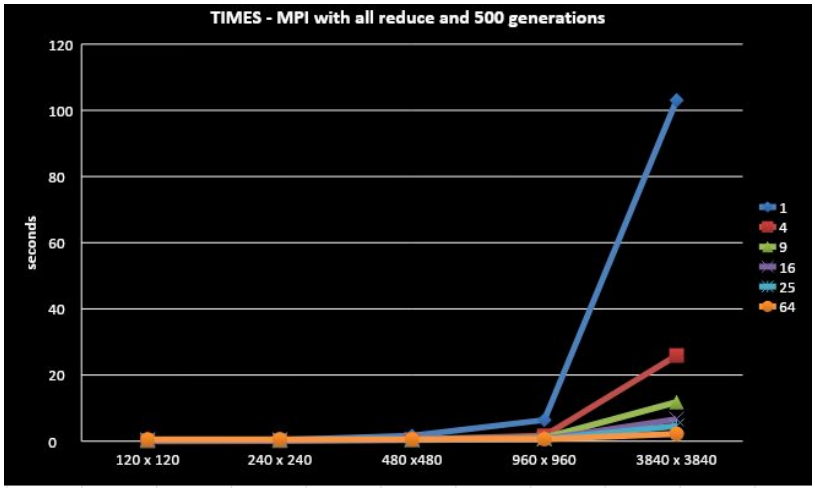
b. MPI πρόγραμμα with allreduce

Τρέχοντας το πρόγραμμα για μεγάλα grid και 500 generations παρατηρήσαμε ότι σπάνια πέφτει στην περίπτωση όπου έχουμε μηδένισμο του ή να παραμένει ίδιο. Έτσι οι χρόνοι είναι παρόμοιοι με αυτούς χωρίς reduce. Βέβαια εκτελέσαμε το πρόγραμμα για μέγεθος <15 και παρατηρήσαμε σημαντική μείωση στον χρόνο εκτέλεσης με allreduce, διότι το grid πιο εύκολα μηδενιζόταν ή παρέμενε το ίδιο πριν από τα 500 generations.

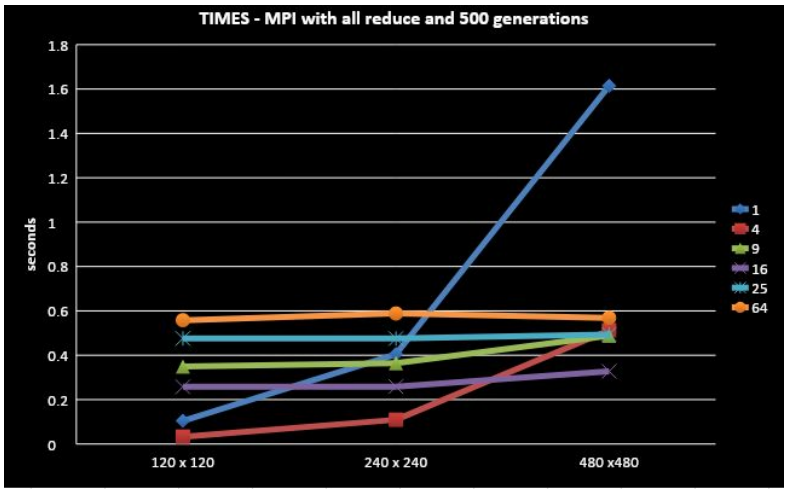
i. Χρόνοι

	MPI with all reduce and 500 generations					
grid\processes	1	4	9	16	25	64
120 x 120	0,104276	0,033134	0,34899	0,257754	0,475733	0,55815
240 x 240	0,405056	0,10938	0,363999	0,258879	0,475248	0,588258
480 x480	1,614466	0,512816	0,487607	0,327602	0,493597	0,568578
960 x 960	6,412040	1,628167	0,998386	0,629241	0,682033	0,629888
3840 x 3840	103,05876	25,89646	11,820829	6,68878	4,58223	2,214798

Το 1ο διάγραμμα παρουσιάζει τους χρονους εκτέλεσης MPI with allreduce για 500 generations, και το 2ο το ίδιο αλλά για τα μικρότερα grids μόνο.



Και σε αυτήν την περίπτωση παρατηρούμε ότι ο χρόνος εκτέλεσης για μεγάλα grid μειώνεται σημαντικά όσο αυξάνεται ο αριθμος των processes.



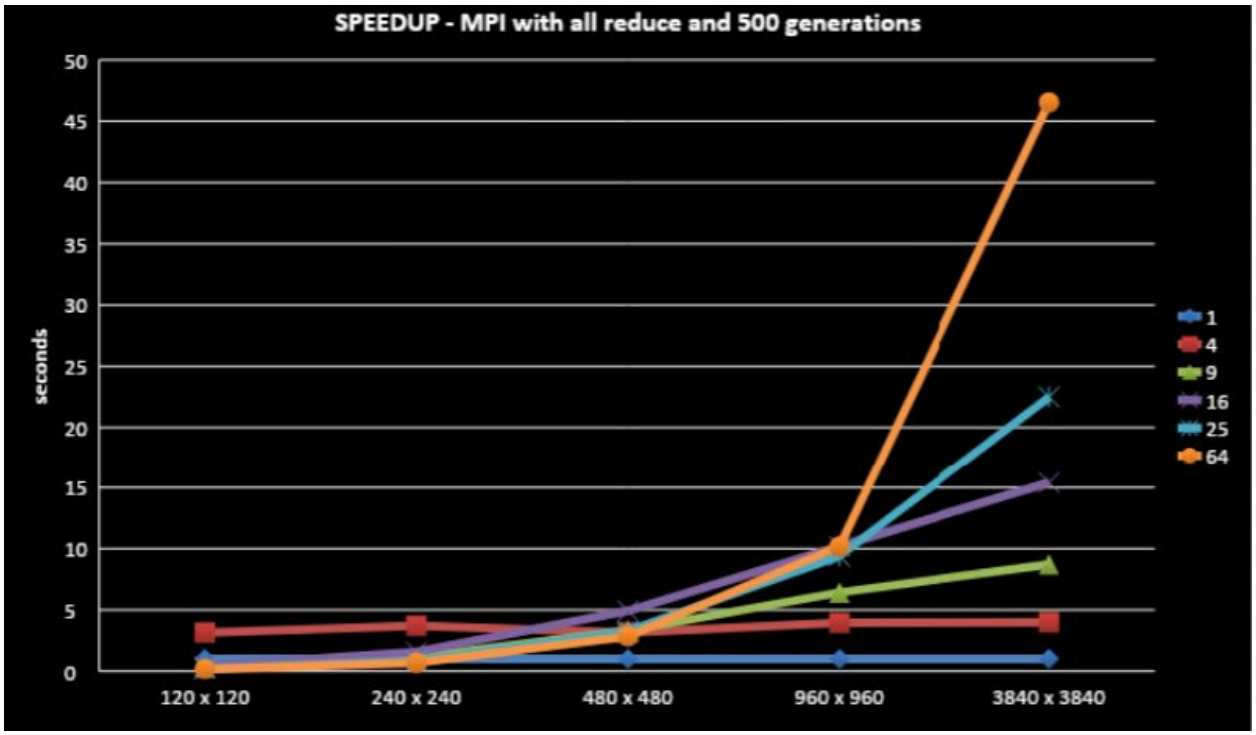
Επίσης και εδώ παρατηρούμε ότι για μικρά grids δεν παρατηρείται σημαντική βελτίωση στον χρόνο εκτέλεσης.

ii. Επιτάχυνση

Επιτάχυνση για MPi with allreduce για 500 generations

Επιτάχυνση S(side,p) = Ts(side) / T(side,p)

	MPI with all reduce and 500 generations					
grid\processes	1	4	9	16	25	64
120 x 120	1	3,147099656	0,2987936617	0,4045562823	0,219190176	0,1868243304
240 x 240	1	3,703199854	1,112794266	1,564653757	0,852304481	0,6885686213
480 x 480	1	3,148236404	3,310998407	4,928132307	3,270818097	2,839480247
960 x 960	1	3,938195529	6,422405763	10,19011794	9,401363277	10,17965099
3840 x 3840	1	3,979646639	8,71840376	15,40770664	22,49096182	46,53190043

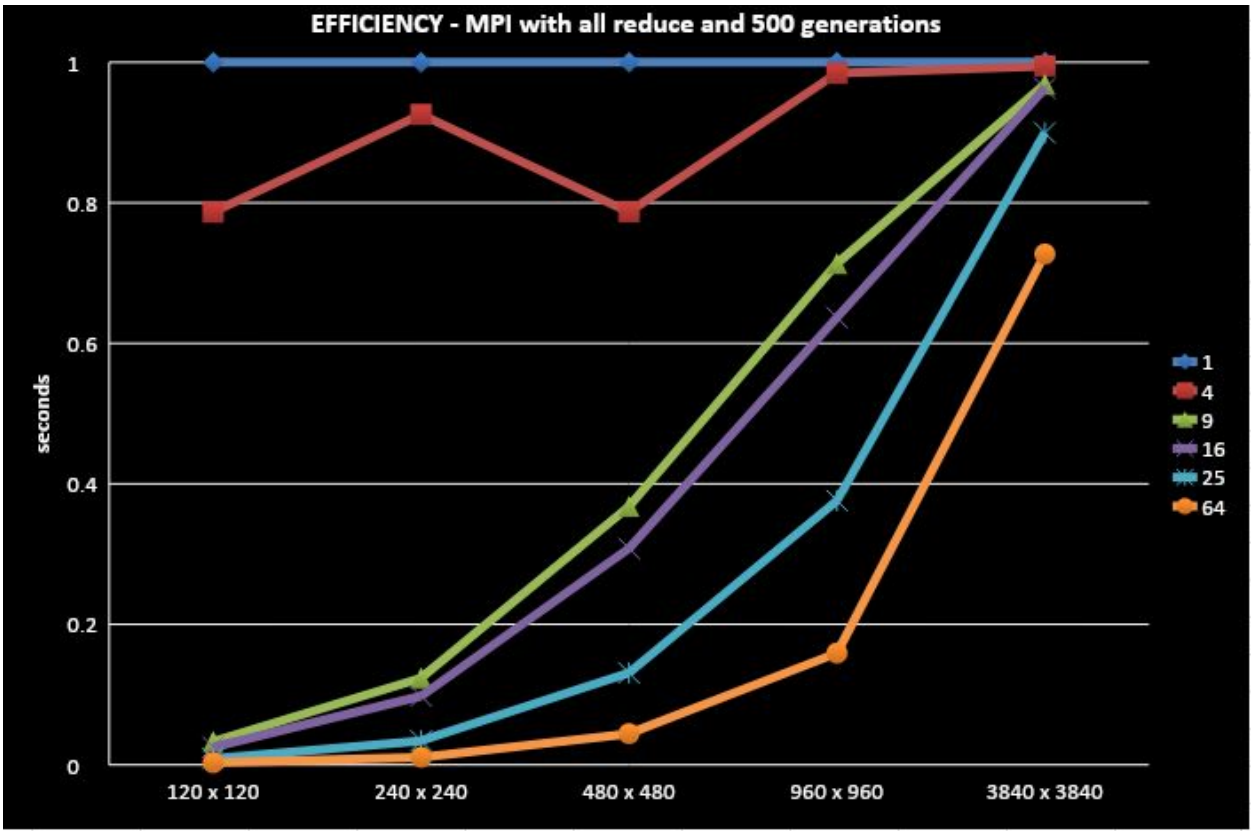


Παρόμοια με την επιταχυνση χωρίς allreduce. Σε μικρά grid λιγοτερα processes έχουν μεγαλύτερη επιτάχυνση, ενώ σε μεγαλύτερα τα πολλά processes παρουσιάζουν σημαντική επιτάχυνση.

iii. Αποδοτικότητα

Αποδοτικότητα για MPI with allreduce και 500 generations

Αποδοτικότητα	MPI with all reduce and 500 generations						
Ε(side,p)=S(side,p)/p	grid\processes	1	4	9	16	25	64
	120 x 120	1	0,786774914	0,03319929575	0,02528476765	0,00876760704	0,002919130162
	240 x 240	1	0,9257999634	0,1236438073	0,09779085982	0,03409217924	0,01075888471
	480 x 480	1	0,7870591011	0,3678887118	0,3080082692	0,1308327239	0,04436687886
	960 x 960	1	0,9845488823	0,7136006403	0,636882371	0,3760545311	0,1590570466
	3840 x 3840	1	0,9949116597	0,9687115289	0,9629816648	0,899638473	0,7270609442



Και σε αυτήν την περίπτωση η αποδοτικότητα είναι καλύτερη για μικρότερο αριθμό διεργασιών για αυτά τα grids.

c. MPI + allreduce + OpenMp για παραλληλοποίηση υπολογισμών

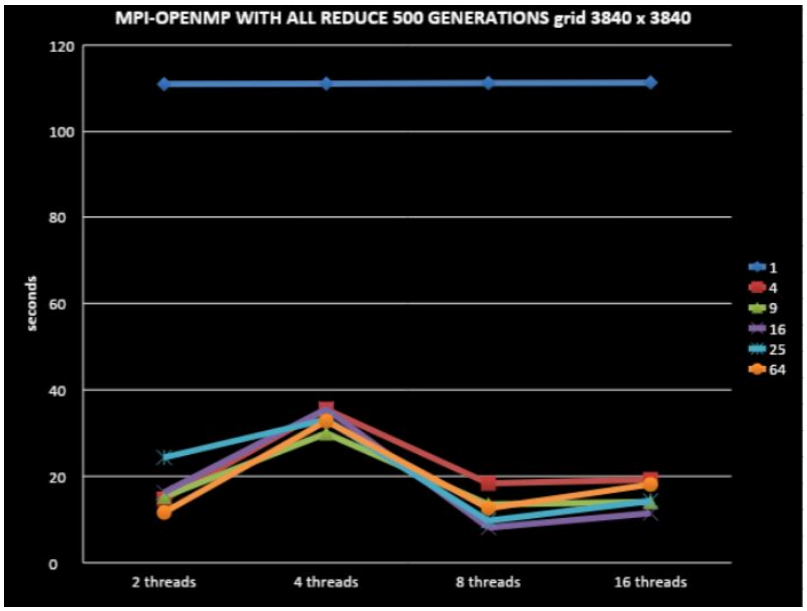
Ενσωμάτωση στο MPI+allreduce εντολές OpenMp για παραλληλοποίηση υπολογισμών.

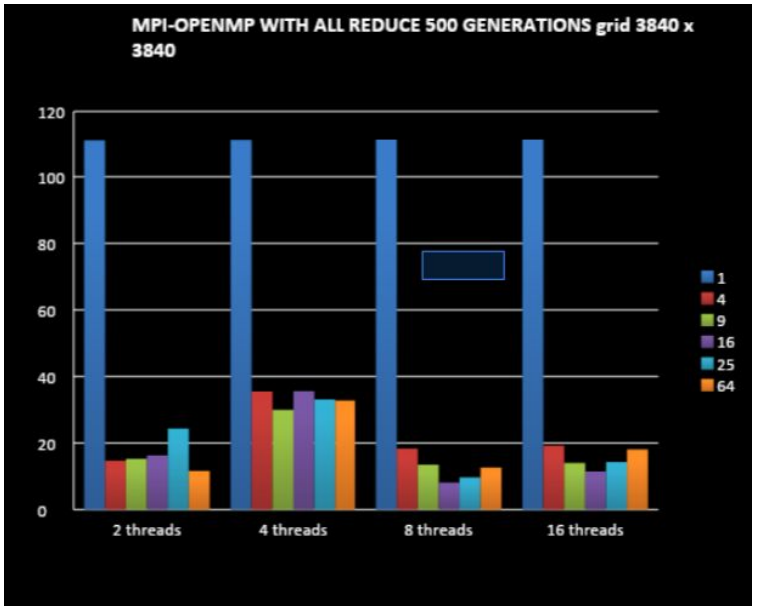
Τρέξαμε αρχικά πολλούς συνδυασμούς διεργασιών και νημάτων για μεγάλο grid και βρήκαμε τους καλύτερους και αυτούς χρησιμοποιούμε για τις υπόλοιπες μετρήσεις.

i. Διεργασίες και νήματα για grid μεγέθους 3840x3840

MPI-OPENMP WITH ALL REDUCE 500 GENERATIONS grid 3840 x 3840						
threads\processes	1	4	9	16	25	64
2 threads	110,995536	14,737867	15,297874	16,291554	24,383072	11,649561
4 threads	111,094379	35,535231	29,973876	35,603143	33,152611	32,79197
8 threads	111,210873	18,365067	13,52851	8,137670	9,719637	12,653414
16 threads	111,331767	19,25735	14,056161	11,47939	14,331633	18,114864

Ο παραπάνω πίνακας και τα παρακάτω διαγράμματα απεικονίζουν τα αποτελέσματα χρόνου εκτέλεσης του προγράμματος MPI-OPENMP with allreduce για 500 generations, για το grid 3840x3840. Αυτήν την μελέτη την κάναμε, ώστε να παρατηρήσουμε πως συμπεριφέρεται το πρόγραμμα για το ίδιο μέγεθος προβλήματος, αλλά για κάθε διεργασία με διαφορετικό αριθμό νημάτων κάθε φορά, και να αποφανθούμε ποιά είναι η καλύτερη επιλογή για αριθμό νημάτων, ανάλογα με το πόσα processes χρησιμοποιούμε.





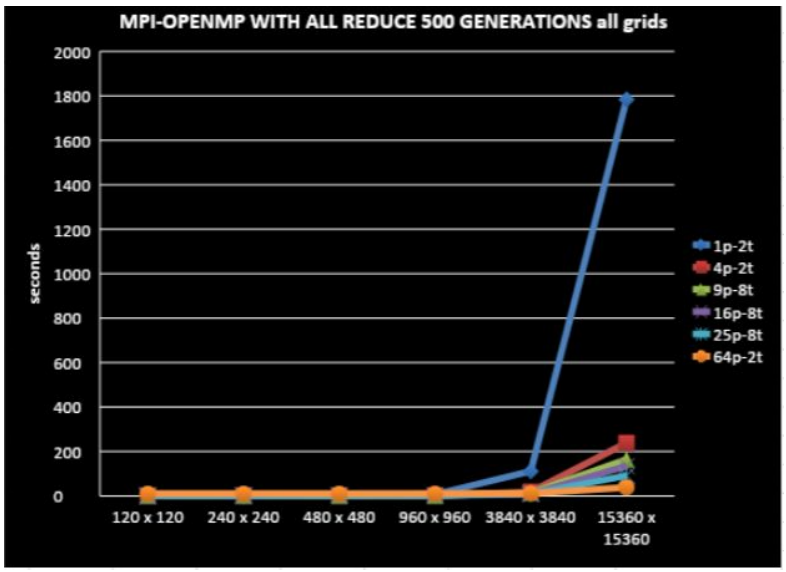
Παρατηρούμε οτι για οι μικρότεροι χρόνοι εκτέλεσης προκύπτουν για τους εξής συνδυασμούς διεργασιών και νημάτων: 1 διεργασία και 2 νήματα, 4 διεργασίες και 2 νήματα, 9 διεργασιες και 8 νήματα, 16 διεργασίες και 8 νήματα, 25 διεργασίες και 8 νήματα και 64 διεργασίες και 2 νήματα.

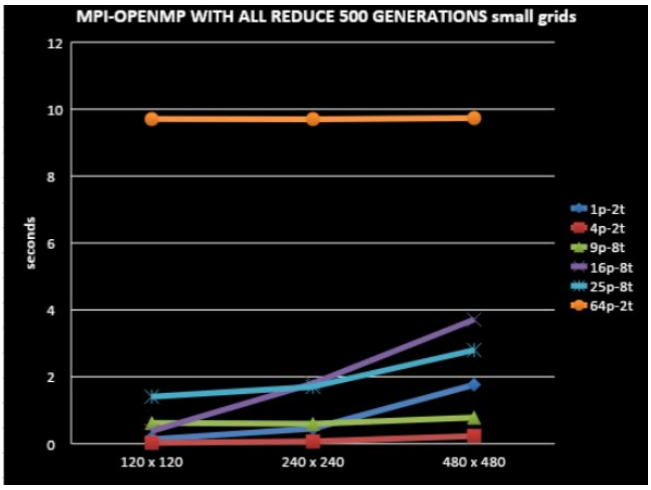
Έτσι για τα υπόλοιπα μεγέθη grid κρατήσαμε αυτούς τους συνδυασμούς για την μελέτη μας.

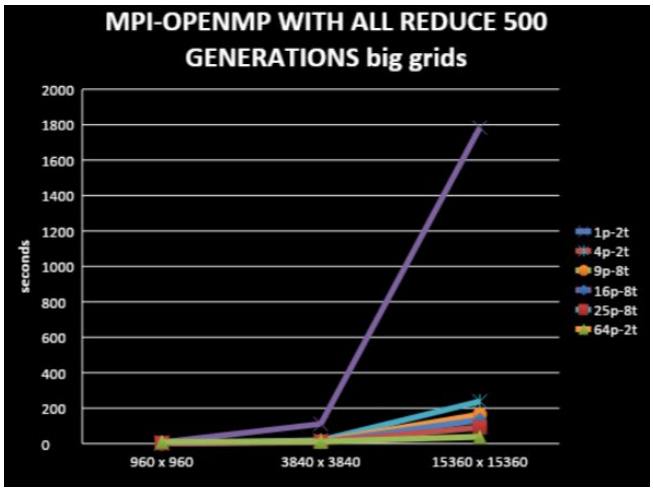
ii. Χρόνοι για τα grids

MPI-OPENMP WITH ALL REDUCE 500 GENERATIONS all grids						
grid \ (procceses-threads)	1p-2t	4p-2t	9p-8t	16p-8t	25p-8t	64p-2t
120 x 120	0,1304747	0,027983	0,624798	0,38025	1,406752	9,702201
240 x 240	0,453873	0,068151	0,596703	1,803357	1,703349	9,701799
480 x 480	1,762461	0,22978	0,785339	3,712122	2,799551	9,732612
960 x 960	6,937716	0,923698	1,737378	2,837543	2,80379	9,908741
3840 x 3840	110,995536	18,365067	13,52851	8,137670	9,719637	11,649561
15360 x 15360	1.783,246806	238,490058	164,099833	133,994583	91,212691	38,441104

Τα παρακάτω διαγράμματα απεικονίζουν τον πίνακα, απλώς με διαφορετικό αριθμό grids, ώστε να φανούν καλύτερα οι διαφορές.Το 1ο τα έχει όλα, το 2ο τα μικρά, το







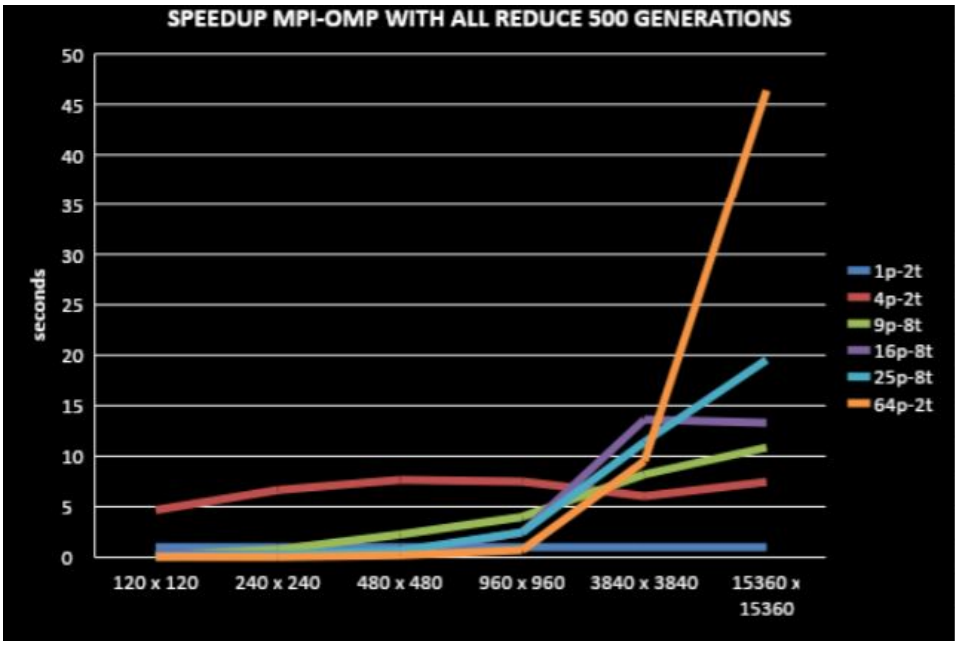
Για μικρά μεγεθη grid ο συνδυασμός μικρού αριθμού processes και νηματων είναι πιο γρήγορος από ότι ο μεγάλος αριθμός διεργασιών με νήματα. Για μεγάλα grid παρατηρούμε ότι έχουμε σημαντική βελτίωση στον χρόνο εκτέλεσης για συνδυασμό με μεγάλο αριθμό διεργασιών. Ουσιαστικά παρατηρούμε παρόμοια κλιμάκωση με το MPI πρόγραμμα.

Βέβαια σε αυτήν την περίπτωση προσθέσαμε και τον πίνακα μεγέθους 15360x15360 για να δούμε πως συμπεριφέρεται το πρόγραμμα και σε αυτήν την περίπτωση. Παρατηρούμε ότι οι χρόνοι με περισσότερα processes και threads είναι αισθητά μικρότεροι σε σχέση με τα λιγότερα. Ουσιαστικά για αυτό το μεγάλο grid, πολυπλοκότητα των for, μειώνεται σε μεγάλο βαθμό.

iii. Επιτάχυνση

Επιτάχυνση S(side,p) = Ts(side) / T(side,p)

SPEEDUP	MPI-OPENMP with all reduce and 500 generations					
grid \ (procceses-threads)	1p-2t	4p-2t	9p-8t	16p-8t	25p-8t	64p-2t
120 x 120	1	4.662641604	0.2088270129	0.3431287311	0.09274889959	0.01344794856
240 x 240	1	6.659814236	0.7606346876	0.2516822792	0.266459193	0.04678235449
480 x 480	1	7.670210636	2.244204095	0.4747853115	0.6295513102	0.1810881806
960 x 960	1	7.51080548	3.993210459	2.444972992	2.474406428	0.7001612011
3840 x 3840	1	6.043840515	8.20456473	13.63971948	11.4197203	9.527872853
15360 x 15360	1	7.477237504	10.86684108	13.30834998	19.55042425	46.38906328



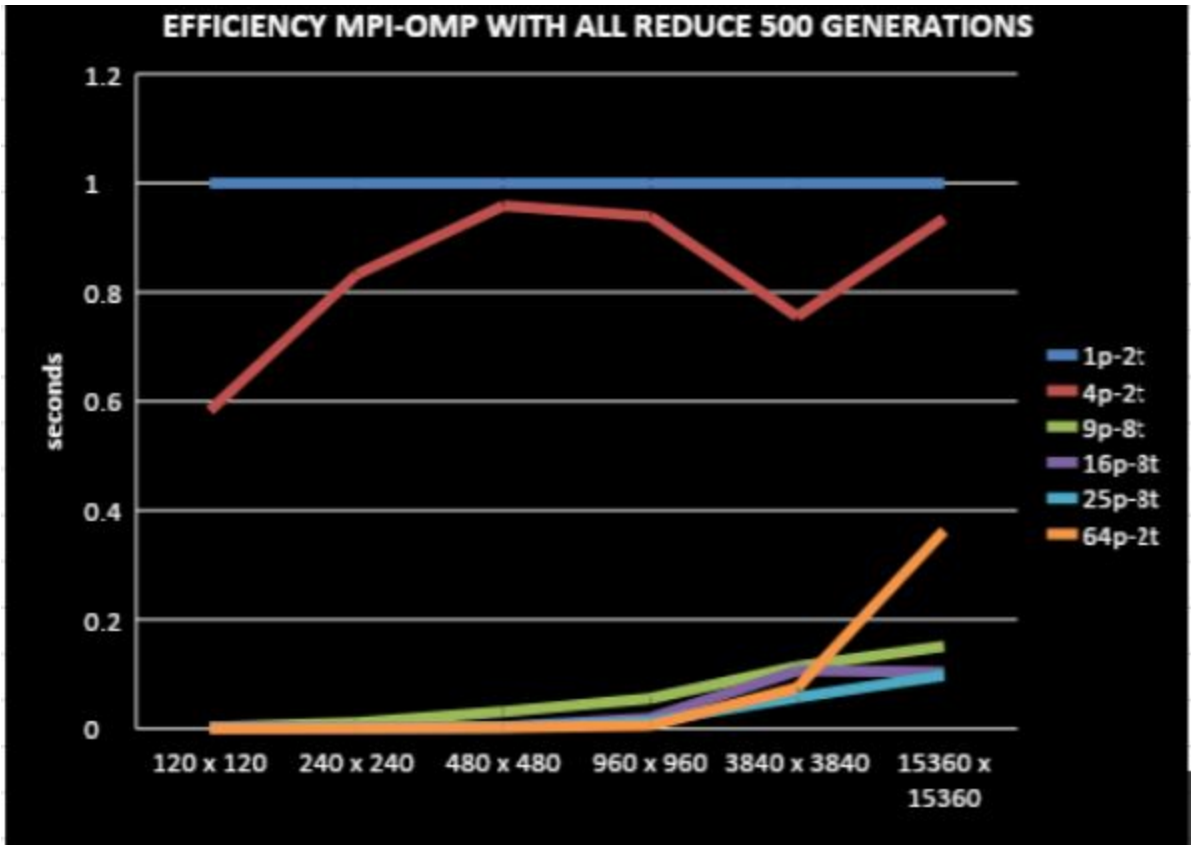
Έχουμε μεγάλη επιτάχυνση για 4 διεργασίες και 2 νήματα για μικρό μέγεθος grid, ενω για μεγάλα μεγέθη η παρατηρείται σημαντική αύξηση της επιτάχυνσης για 64 διεργασίες και 2 νήματα.

iv. Αποδοτικότητα

Αποδοτικότητα	EFFICIENCY	MPI-OPENMP with all reduce and 500 generations					
E(side,p)=S(side,p)/p	grid \ (procceses-threads)	1p-2t	4p-2t	9p-8t	16p-8t	25p-8t	64p-2t
	120 x 120	1	0.5828302005	0.002900375179	0.002680693212	0.000463744498	0.0001050620982
	240 x 240	1	0.8324767795	0.01056437066	0.001966267806	0.001332295965	0.0003654871444
	480 x 480	1	0.9587763295	0.03116950132	0.003709260246	0.003147756551	0.001414751411
	960 x 960	1	0.938850685	0.05546125637	0.0191013515	0.01237203214	0.005470009384
	3840 x 3840	1	0.7554800644	0.1139522879	0.1065603084	0.05709860152	0.07443650666
	15360 x 15360	1	0.934654688	0.1509283483	0.1039714842	0.09775212125	0.3624145569

Στην αρχή στην αποδοτικότητα σκεφτήκαμε να κανουμε την επιτάχυνση προς τον αριθμό των processes, αλλά συνειδητοποιήσαμε ότι έβγαινε μεγαλύτερο από 1 αρχικά, αλλά και δεν είχε και πολύ νόημα, αφού ο αριθμός των νημάτων παίζει αρκετά σημαντικό ρόλο στην εκτέλεση του προγράμματος.

Οπότε καταλήξαμε στο να υπολογίσουμε την αποδοτικότητα διαιρώντας την επιτάχυνση με το {διεργασιες*νήματα}.



Παρατηρούμε οτι για 4 διεργασίες και 2 νήματα έχουμε την μεγαλύτερη αποδοτικότητα για αυτά τα μεγέθη grid. Επίσης παρατηρούμε ότι οι 64 διεργασίες με 2 νήματα, η αποδοτικότητα στο μεγάλο grid έχει αρχίσει ανοδική πορεία.

5. Συμπεράσματα

Παρατηρούμε γενικά ότι όσο μεγαλώνει το grid και ο αριθμός των διεργασιών τόσο μειώνεται και ο χρόνος εκτέλεσης.
Επίσης, για τα grids που χρησιμοποιούμε, στις περισσότερες περιπτώσεις το MPI είναι πιο γρήγορο από την OpenMP υλοποίηση, αν και για 4 processes και 2 threads, έχουμε αρκετά καλούς χρόνους για το OpenMP.
Επιπλέον τρέξαμε το MPI+OpenMP με collapse(2) , και οι χρόνοι ήταν παρόμοιοι για όλες τις περιπτώσεις, οπότε δεν το συμπεριλήφθηκε στην ανάλυση μας.

Συμπερασματικά, καταλήγουμε στο γεγονός ότι η παραλληλοποίηση του προγράμματος, βοηθάει σημαντικά στην μείωση του χρόνου εκτέλεσης σε σχέση με ένα σειριακό, ειδικά στα μεγάλα προβλήματα.

6. Έξοδος mpiP

Η έξοδος του mpiP βρίσκεται στο αρχείο mpiPresult3840grid-64p-500g.txt και μας βοήθησε να δούμε σε ποιες MPI εντολές ξοδεύει περισσότερο χρόνο το πρόγραμμα.