Παράλληλα Συστήματα

Project MPI - MPI/OpenMP – CUDA

Συμμετέχοντες:

Νίκος Σοφράς (Α.Μ.: 1115201200168)

Βαγγέλης Τσιατούρας (Α.Μ.: 1115201200185)

ΠΕΡΙΕΧΟΜΕΝΑ

Παραδοχές

- Όλα τα προγράμματα μεταγλωττίζονται μέσω makefile (έχει υλοποιηθεί separate compilation).
- Τα προγράμματα δουλεύουν με τετράγωνους πίνακες.
- Το πλήθος των processes πρέπει να είναι τέλειο τετράγωνο, π.χ. 1, 4, 9, 16, 25 κλπ.
- Οι υποπίνακες (blocks) στα προγράμματα MPI, MPI/OpenMP είναι τετράγωνοι. Το μέγεθος του αρχικού πίνακα πρέπει να διαιρείται ακριβώς με την τετραγωνική ρίζα του πλήθους των processes. Π.χ. δεν μπορεί να δοθεί μέγεθος πίνακα 1000 και πλήθος διεργασιών 9.
- Το αρχείο εισόδου, άμα δοθεί, πρέπει να περιέχει "0" ή "." για τα νεκρά κελιά και "1" ή "*" για τα ζωντανά. Δίνονται αρχεία στο φάκελο inputs με έτοιμες αρχικές καταστάσεις που χρησιμοποιήσαμε στο testing.
- Όταν δεν δοθεί αρχείο εισόδου παράγεται τυχαία η αρχική γενιά, όπου κάθε κελί έχει πιθανότητα 30% να είναι ζωντανό.
- Παρέχεται η δυνατότητα εκτύπωσης της κάθε γενιάς, όπου κάθε γενιά αποθηκεύεται σε διαφορετικό αρχείο στο φάκελο outputs. Αυτό καθυστερεί την εκτέλεση του προγράμματος, γι' αυτό γίνεται μόνο αν επιλεχθεί από το χρήστη.
- Ο λόγος που επιλέξαμε η αρχικοποίηση του πίνακα να γίνεται μόνο από το process 0 είναι για να διατηρήσουμε τη λειτουργικότητα της εκτύπωσης της κάθε γενιάς.

Ορίσματα γραμμής εντολών

MPI

Εντολή εκτέλεσης:

```
mpiexec -f <machine_file> -n <number_of_processes> ./game_of_life –d
<dimension_of_grid> -f <input_file> -l <number_of_loops> -p
```

-d: διάσταση του πίνακα
-f: αρχείο που περιέχει αρχική κατάσταση
-l: αριθμός επαναλήψεων (γενεών)
-p: εκτύπωση κάθε γενιάς σε αρχείο. (καθυστερεί πολύ την εκτέλεση)

MPI / OpenMP

Εντολή εκτέλεσης:
`mpiexec -f <machine_file> -p <number_of_processes> -bind-to core ./game_of_life -t <number_of_threads> –d <dimension_of_grid> -f <input_file> -l <number_of_loops> -p`

-t: πλήθος threads
-d: διάσταση του πίνακα
-f: αρχείο που περιέχει αρχική κατάσταση
-l: αριθμός επαναλήψεων (γενεών)
-p: εκτύπωση κάθε γενιάς σε αρχείο. (καθυστερεί πολύ την εκτέλεση)

CUDA

Εντολή εκτέλεσης:
`./game_of_life –d <dimension_of_grid> -f <input_file> -l <number_of_loops> -p`

-d: διάσταση του πίνακα
-f: αρχείο που περιέχει αρχική κατάσταση
-l: αριθμός επαναλήψεων (γενεών)
-p: εκτύπωση κάθε γενιάς σε αρχείο. (καθυστερεί πολύ την εκτέλεση)

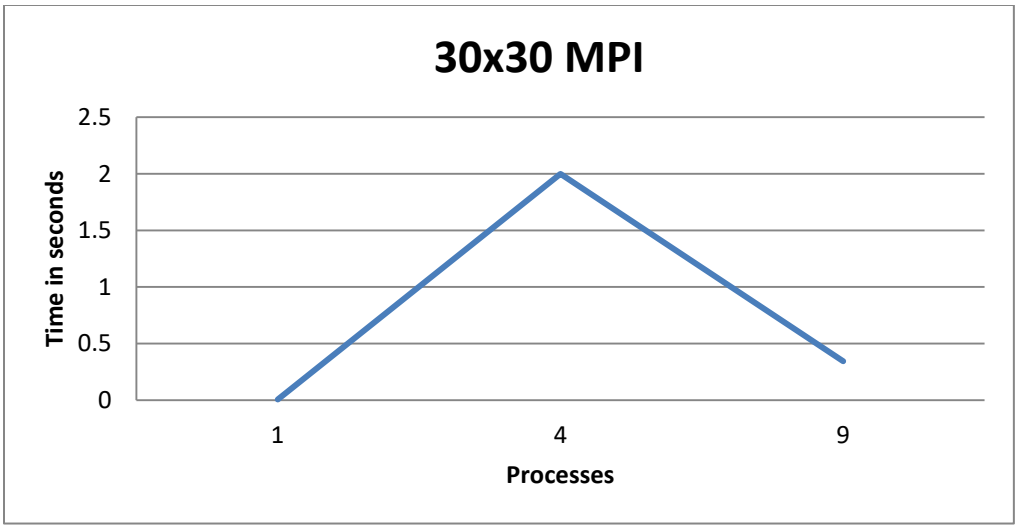
Στατιστικά

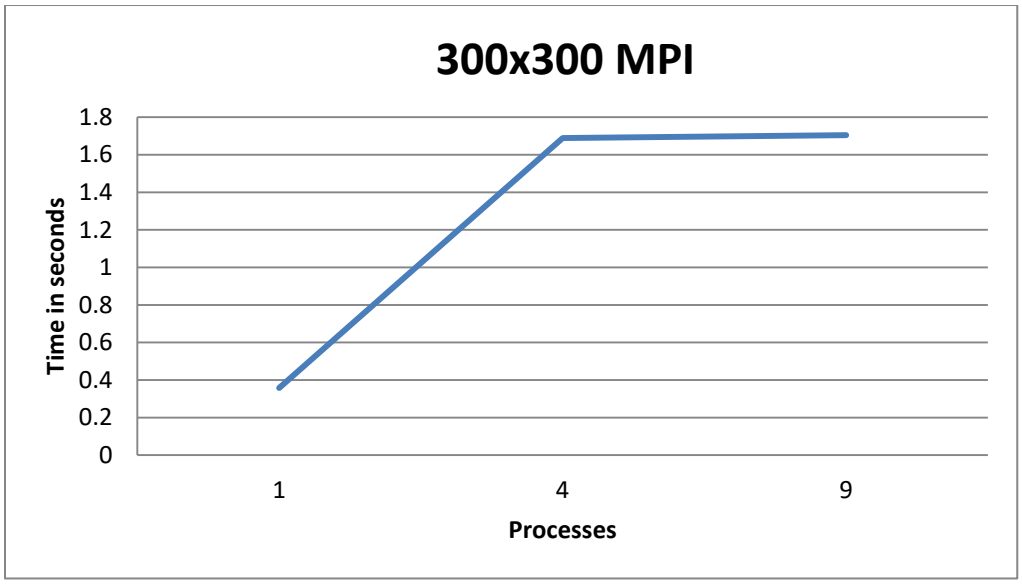
MPI

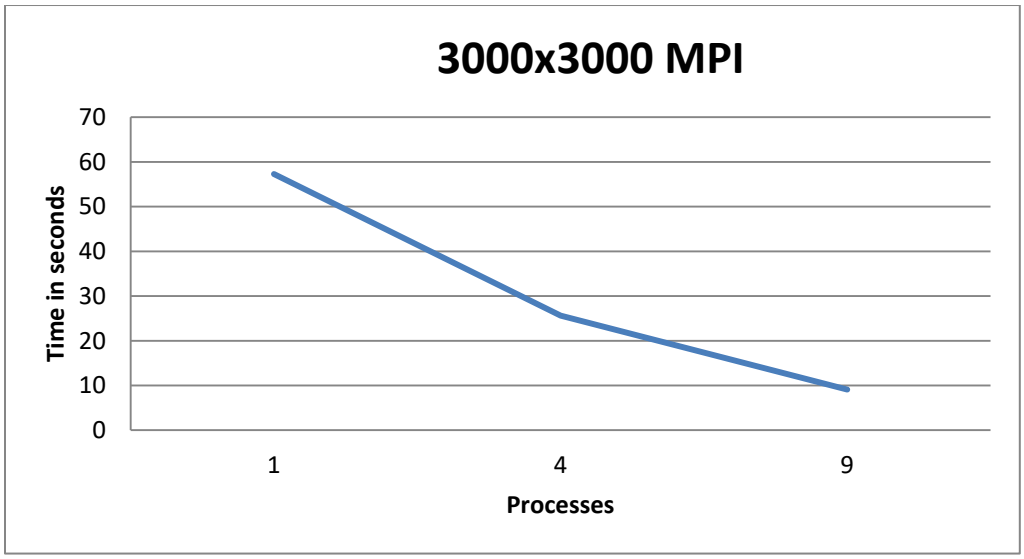
Συνολικά έγιναν 9 εκτελέσεις (μέσω του mpi_bench.sh). Η διαδικασία η οποία ακολουθήθηκε είναι ότι γίνονται κλιμακωτές αυξήσεις στο πλήθος των επεξεργαστών και στις διαστάσεις του πίνακα. Πιο αναλυτικά, οι διαστάσεις του πίνακα είναι 30x30, 300x300, 3000x3000 και το πλήθος των επεξεργαστών είναι 1, 4 και 9. Θα θέλαμε να τρέξουμε και για 30000x30000 διάσταση και 16 επεξεργαστές άλλα οι δυνατότητες του cluster ήταν περιορισμένες, αφού είχαμε στη διάθεση μας μόνο 7 μηχανές.

Ακολουθούν ενδεικτικοί χρόνοι και γραφήματα βασισμένα σε αυτούς.

```
Execution 1
Processors: 1, Dimension: 30x30, Loops: 100
time elapsed: 0.005498 seconds
Terminated successfully

Execution 2
Processors: 1, Dimension: 300x300, Loops: 100
time elapsed: 0.357818 seconds
Terminated successfully

Execution 3
Processors: 1, Dimension: 3000x3000, Loops: 100
time elapsed: 57.297644 seconds
Terminated successfully

Execution 4
Processors: 4, Dimension: 30x30, Loops: 100
time elapsed: 1.999970 seconds
Terminated successfully

Execution 5
Processors: 4, Dimension: 300x300, Loops: 100
time elapsed: 1.689279 seconds
Terminated successfully

Execution 6
Processors: 4, Dimension: 3000x3000, Loops: 100
time elapsed: 25.617679 seconds
Terminated successfully

Execution 7
Processors: 9, Dimension: 30x30, Loops: 100
time elapsed: 0.344268 seconds
Terminated successfully

Execution 8
Processors: 9, Dimension: 300x300, Loops: 100
time elapsed: 1.703980 seconds
Terminated successfully

Execution 9
Processors: 9, Dimension: 3000x3000, Loops: 100
time elapsed: 9.090675 seconds
Terminated successfully
```


30x30 MPI
Time in seconds
2.5
2
1.5
1
0.5
0
1
4
9
Processes

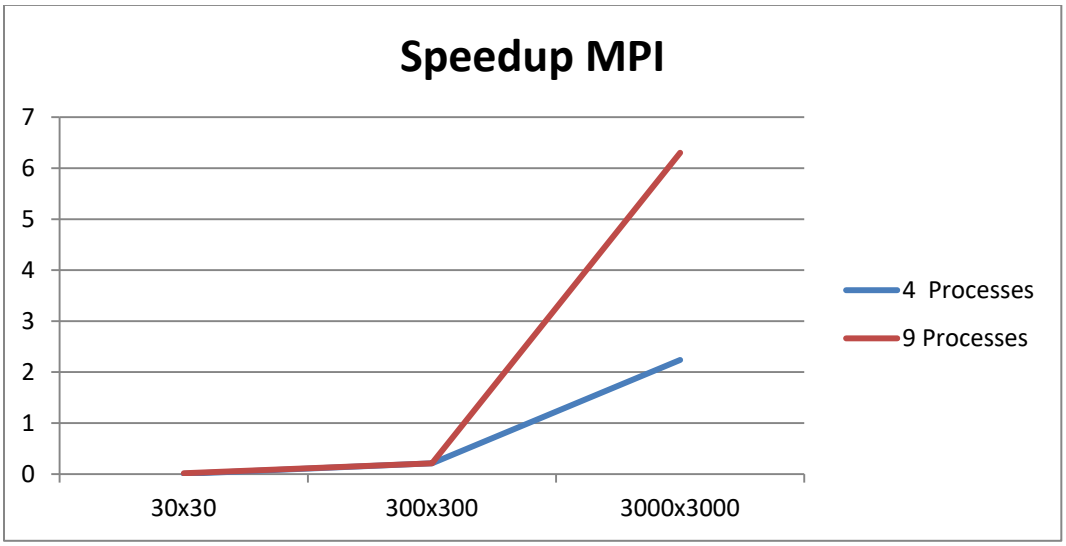

300x300 MPI
Time in seconds
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
4
9
Processes

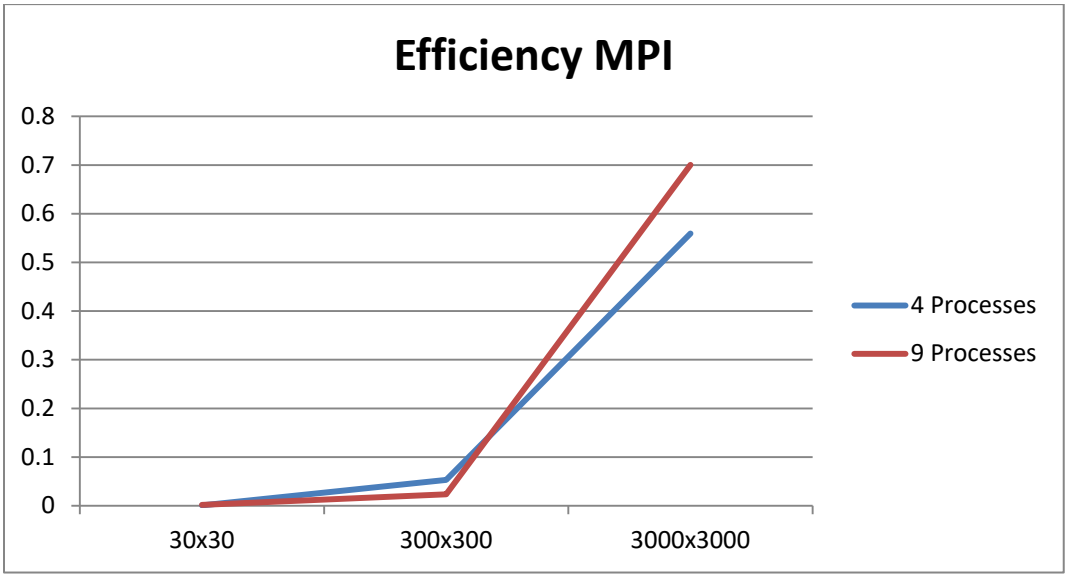

3000x3000 MPI
Time in seconds
70
60
50
40
30
20
10
0
1
4
9
Processes

Παρατηρούμε ότι δεν υπάρχει βελτίωση για διαστάσεις 30x30 και 300x300 για 4 και 9 processes. Αυτό συμβαίνει γιατί το πλήθος των υπολογισμών είναι σχετικά μικρό και το κόστος ανταλλαγής μηνυμάτων επισκιάζει την επιτάχυνση των επιπλέον επεξεργαστών. Για διαστάσεις 3000x3000 παρατηρήθηκε σημαντική βελτίωση στο χρόνο εκτέλεσης αφού το πλήθος των υπολογισμών είναι μεγαλύτερο και το κόστος επικοινωνίας δεν είναι πλέον σημαντικό σε τέτοιο όγκο δεδομένων.

Για τον υπολογισμό των speedup και efficiency χρησιμοποιήθηκαν οι εξής τύποι:

```
Speedup :=  S = Tserial / Tparallel
Efficiency := S / p , ( p = number of processes)
```





MPI / OpenMP

Για την μελέτη στο υβριδικό πρόγραμμα MPI/OpenMP ακολουθήθηκε ίδια τακτική όπως προηγουμένως. Αξίζει να σημειωθεί ότι στο machines host file αντιστοιχήθηκε κάθε μηχανή σε 1 slot αντί για 2. Με αυτό τον τρόπο οι πυρήνες κάθε μηχανήματος είναι διαθέσιμοι να εκτελούνται περισσότερα threads παράλληλα.

Ακολουθούν ενδεικτικοί χρόνοι και γραφήματα βασισμένα σε αυτούς.

2 Threads

```
Execution 1
Processors: 1, Dimension: 30x30, Loops: 100
time elapsed: 0.014811 seconds
Terminated successfully

Execution 2
Processors: 1, Dimension: 300x300, Loops: 100
time elapsed: 0.953001 seconds
Terminated successfully

Execution 3
Processors: 1, Dimension: 3000x3000, Loops: 100
time elapsed: 55.900963 seconds
Terminated successfully

Execution 4
Processors: 4, Dimension: 30x30, Loops: 100
time elapsed: 0.090671 seconds
Terminated successfully

Execution 5
Processors: 4, Dimension: 300x300, Loops: 100
time elapsed: 0.320948 seconds
Terminated successfully

Execution 6
Processors: 4, Dimension: 3000x3000, Loops: 100
time elapsed: 18.764648 seconds
Terminated successfully

Execution 7
Processors: 9, Dimension: 30x30, Loops: 100
time elapsed: 0.328105 seconds
Terminated successfully

Execution 8
Processors: 9, Dimension: 300x300, Loops: 100
time elapsed: 0.369004 seconds
Terminated successfully
```

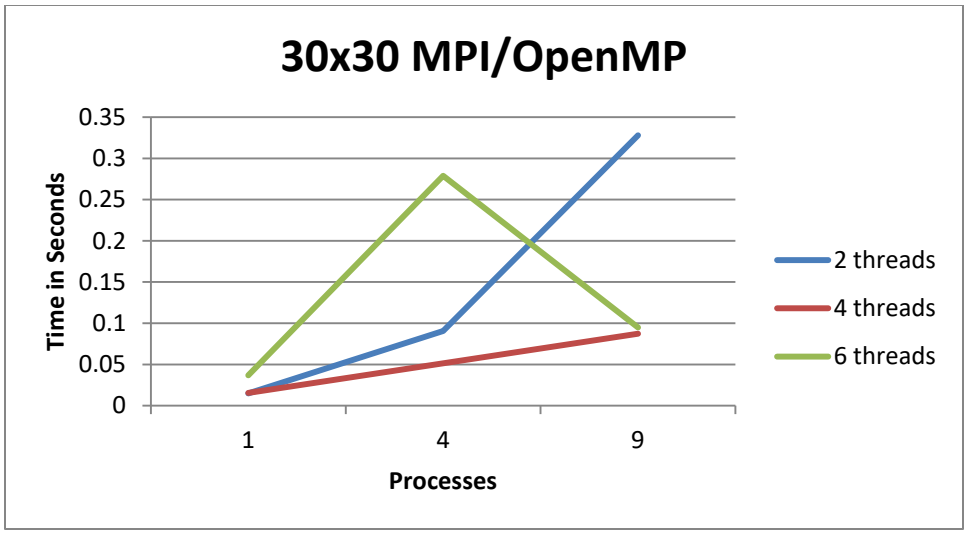
```
Execution 9
Processors: 9, Dimension: 3000x3000, Loops: 100
time elapsed: 9.120810 seconds
Terminated successfully
```

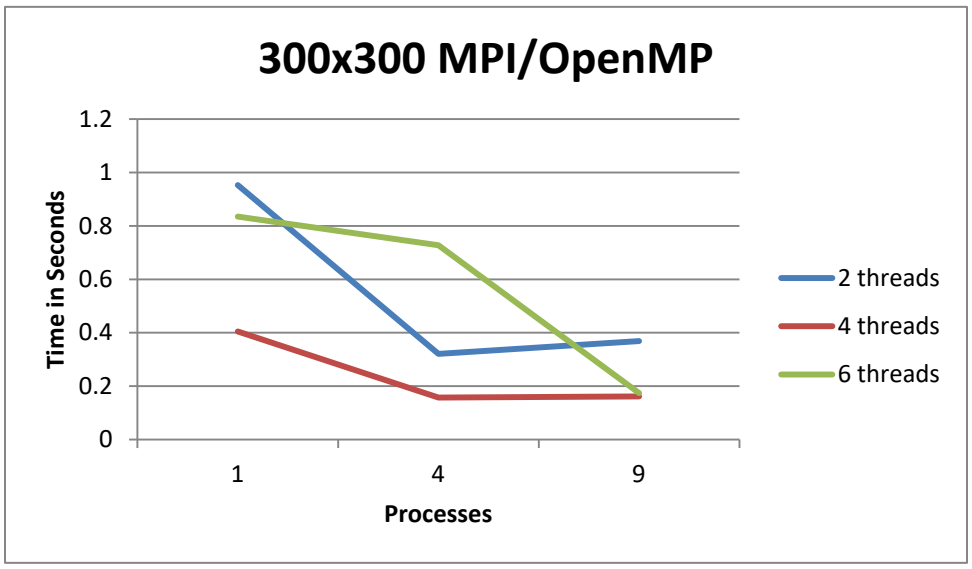
4 Threads

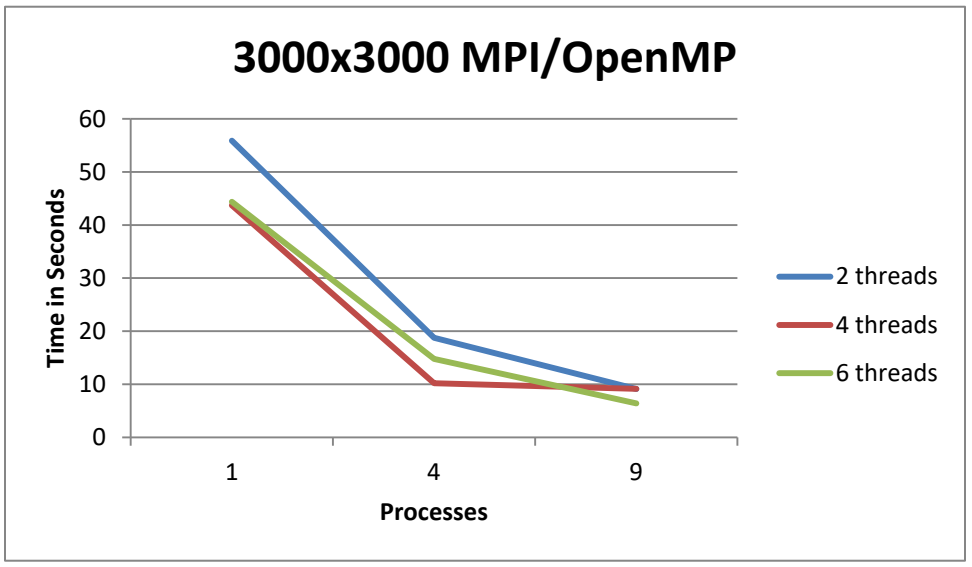
```
Execution 1
Processors: 1, Dimension: 30x30, Loops: 100
time elapsed: 0.015521 seconds
Terminated successfully

Execution 2
Processors: 1, Dimension: 300x300, Loops: 100
time elapsed: 0.404844 seconds
Terminated successfully

Execution 3
Processors: 1, Dimension: 3000x3000, Loops: 100
time elapsed: 43.701195 seconds
Terminated successfully

Execution 4
Processors: 4, Dimension: 30x30, Loops: 100
time elapsed: 0.051321 seconds
Terminated successfully

Execution 5
Processors: 4, Dimension: 300x300, Loops: 100
time elapsed: 0.157564 seconds
Terminated successfully

Execution 6
Processors: 4, Dimension: 3000x3000, Loops: 100
time elapsed: 10.206246 seconds
Terminated successfully

Execution 7
Processors: 9, Dimension: 30x30, Loops: 100
time elapsed: 0.087302 seconds
Terminated successfully

Execution 8
Processors: 9, Dimension: 300x300, Loops: 100
time elapsed: 0.161365 seconds
Terminated successfully

Execution 9
Processors: 9, Dimension: 3000x3000, Loops: 100
time elapsed: 9.140776 seconds
Terminated successfully
```

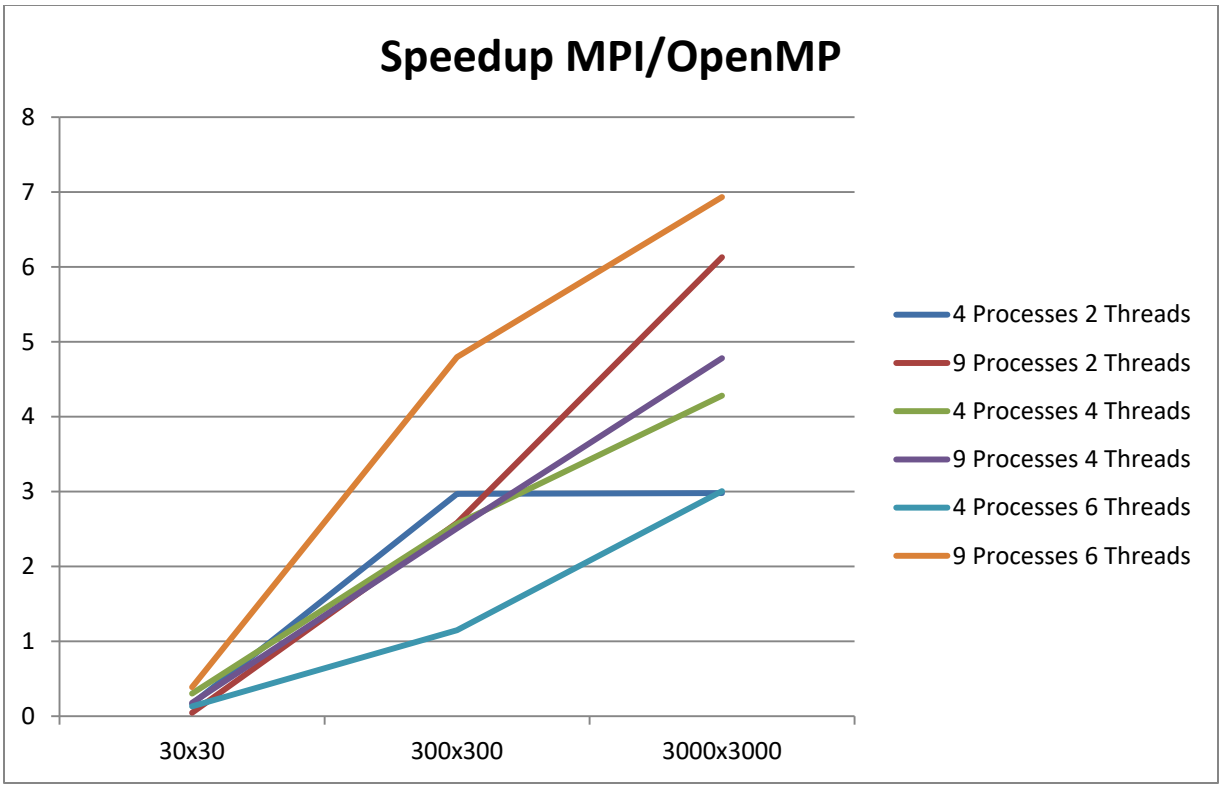
6 Threads

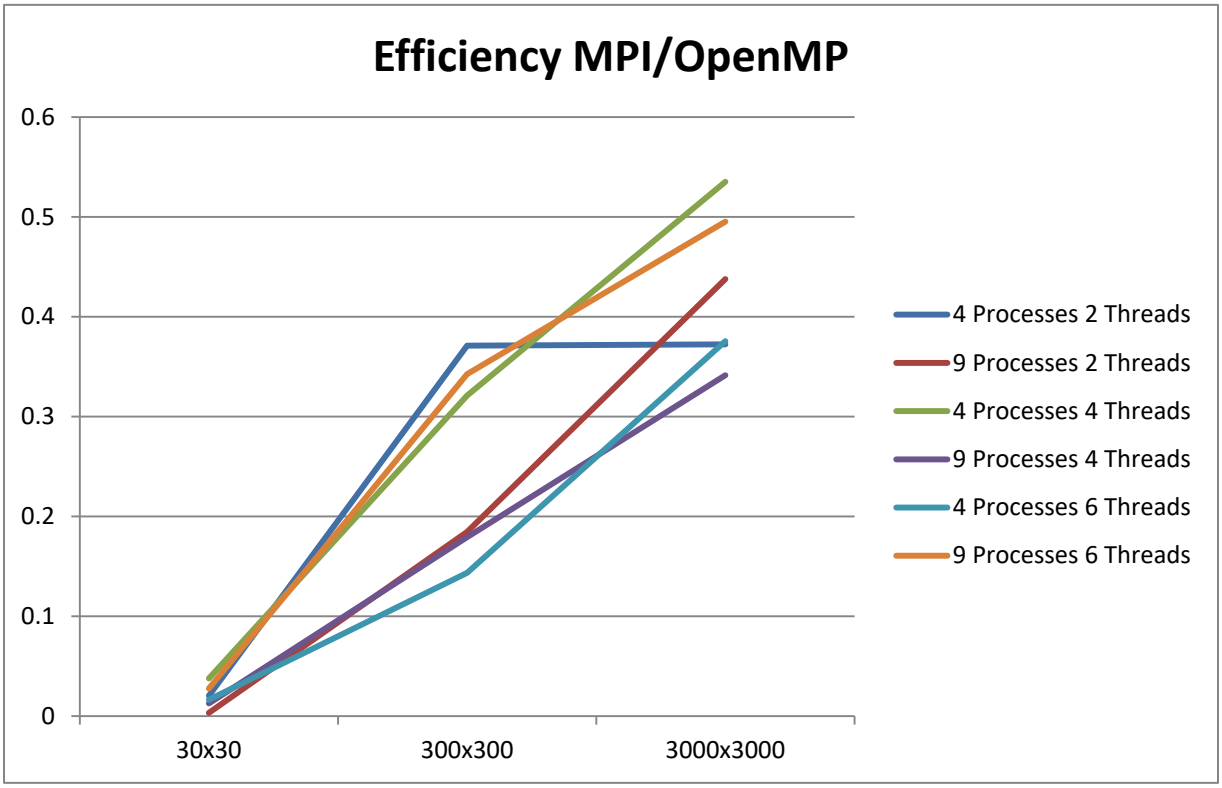
```
Execution 1
Processors: 1, Dimension: 30x30, Loops: 100
time elapsed: 0.036709 seconds
Terminated successfully

Execution 2
Processors: 1, Dimension: 300x300, Loops: 100
time elapsed: 0.835042 seconds
Terminated successfully

Execution 3
Processors: 1, Dimension: 3000x3000, Loops: 100
time elapsed: 44.381509 seconds
Terminated successfully

Execution 4
Processors: 4, Dimension: 30x30, Loops: 100
time elapsed: 0.278834 seconds
Terminated successfully

Execution 5
Processors: 4, Dimension: 300x300, Loops: 100
time elapsed: 0.727574 seconds
Terminated successfully

Execution 6
Processors: 4, Dimension: 3000x3000, Loops: 100
time elapsed: 14.771784 seconds
Terminated successfully

Execution 7
Processors: 9, Dimension: 30x30, Loops: 100
time elapsed: 0.094897 seconds
Terminated successfully

Execution 8
Processors: 9, Dimension: 300x300, Loops: 100
time elapsed: 0.174112 seconds
Terminated successfully

Execution 9
Processors: 9, Dimension: 3000x3000, Loops: 100
time elapsed: 6.400862 seconds
Terminated successfully
```


30x30 MPI/OpenMP
Time in Seconds
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
4
9
Processes
2 threads
4 threads
6 threads


300x300 MPI/OpenMP
Time in Seconds
1.2
1
0.8
0.6
0.4
0.2
0
1
4
9
Processes
2 threads
4 threads
6 threads


3000x3000 MPI/OpenMP
Time in Seconds
60
50
40
30
20
10
0
1
4
9
Processes
2 threads
4 threads
6 threads


Speedup MPI/OpenMP
8
7
6
5
4
3
2
1
0
30x30
300x300
3000x3000
4 Processes 2 Threads
9 Processes 2 Threads
4 Processes 4 Threads
9 Processes 4 Threads
4 Processes 6 Threads
9 Processes 6 Threads

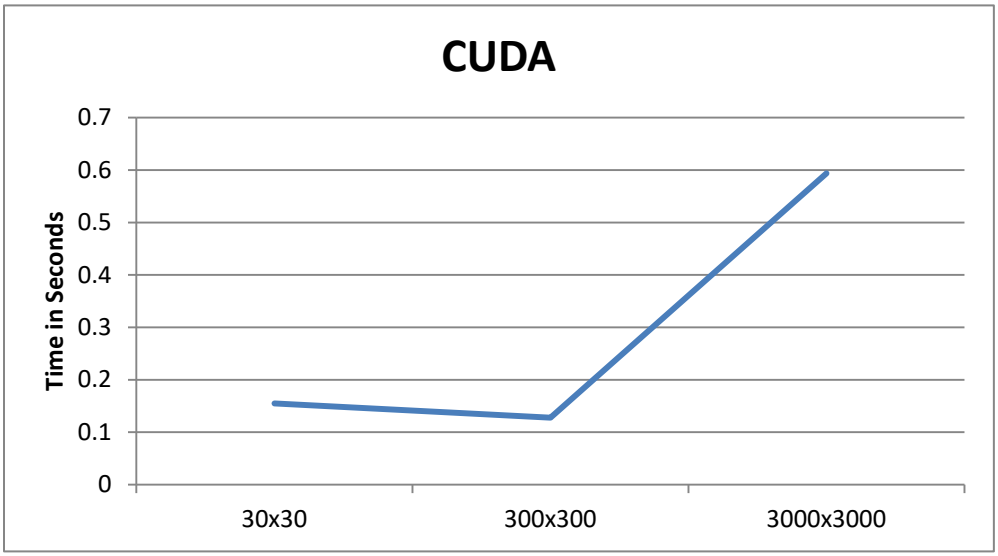

Efficiency MPI/OpenMP
0.6
0.5
0.4
0.3
0.2
0.1
0
30x30
300x300
3000x3000
4 Processes 2 Threads
9 Processes 2 Threads
4 Processes 4 Threads
9 Processes 4 Threads
4 Processes 6 Threads
9 Processes 6 Threads

CUDA

Λόγω των τεχνικών προβλημάτων η υλοποίηση του CUDA δοκιμάστηκε σε προσωπικό υπολογιστή που περιέχει GPU Nvidia GTX 960 4GB. Επίσης παρατηρήθηκε σημαντική βελτίωση στον χρόνο εκτέλεσης σε σχέση με τις προηγούμενες υλοποιήσεις (MPI-MPI/OpenMP) διότι γίνεται χρήση περισσότερων threads (μερικές εκατοντάδες) σε σχέση με τα 4 και 9 που δοκιμάστηκαν. Ακόμη μεγάλη βαρύτητα στο χρόνο εκτέλεσης έδωσε η μνήμη της κάρτας γραφικών που είναι κοινή για όλα τα threads καθώς ο χρόνος επικοινωνίας μεταξύ τους μειώθηκε σημαντικά.

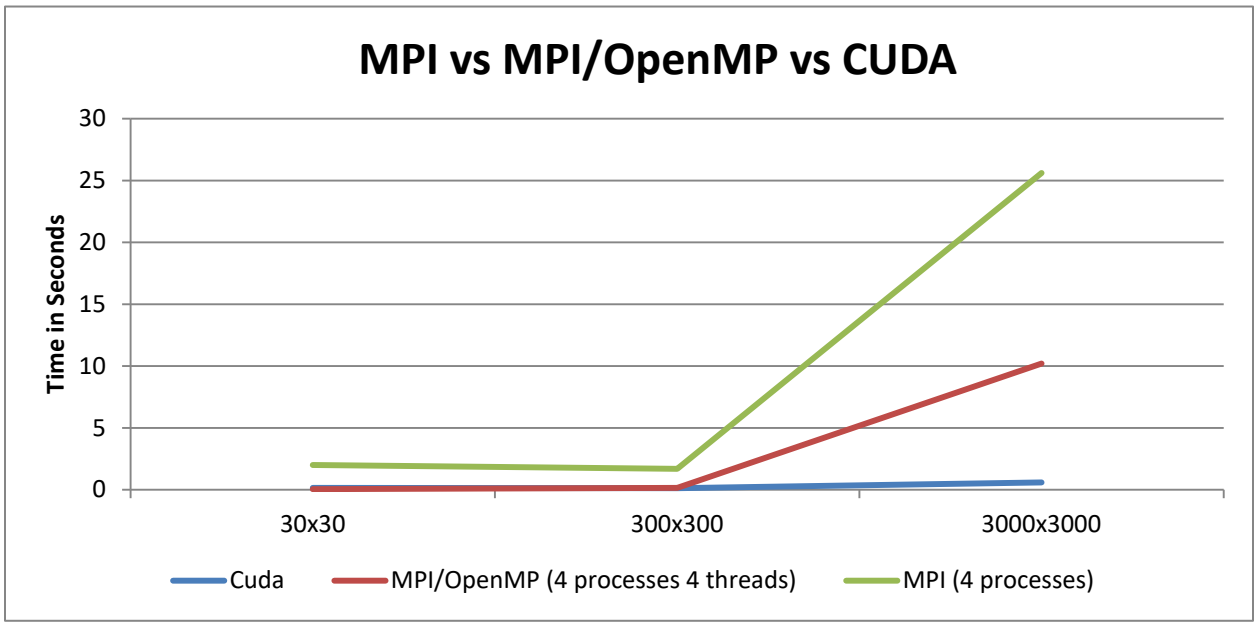
Ακολουθούν ενδεικτικοί χρόνοι και γραφήματα βασισμένα σε αυτούς.

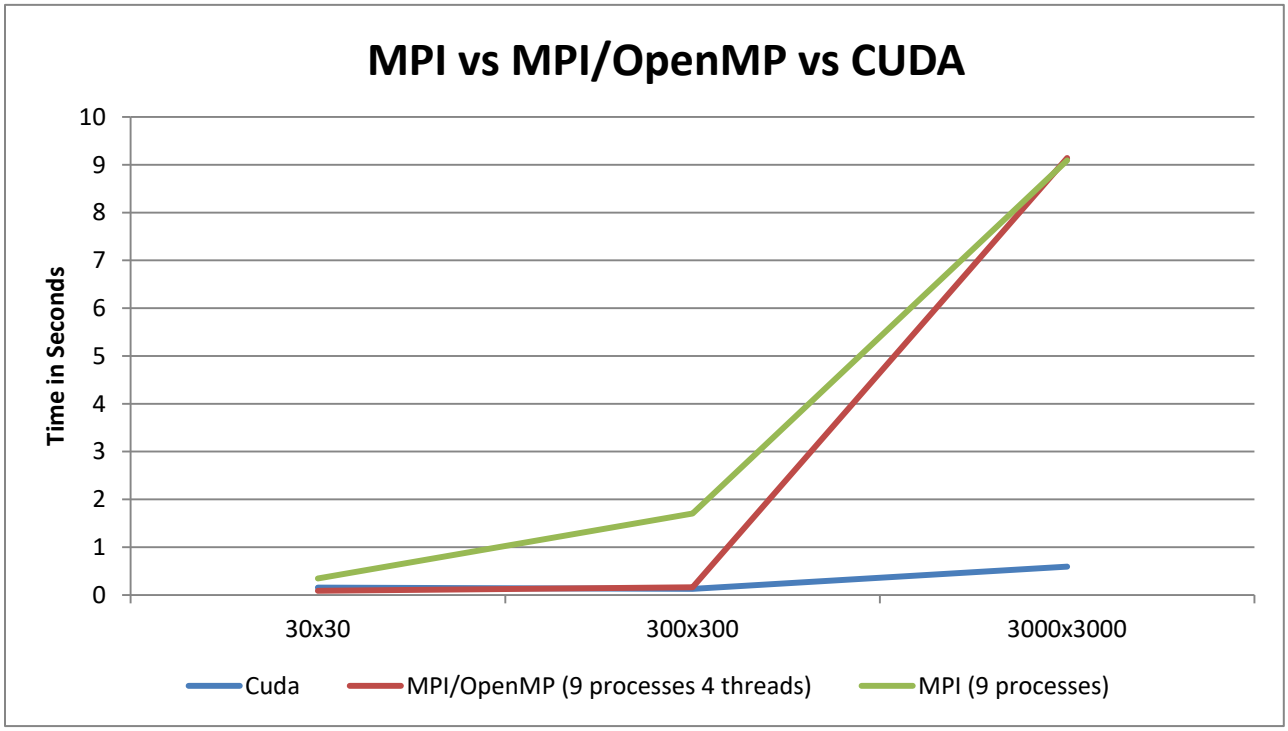
```
Execution 1
Processors: 4, Dimension: 30x30, Loops: 100
time elapsed: 0.154774
Terminated successfully

Execution 2
Processors: 4, Dimension: 300x300, Loops: 100
time elapsed: 0.127811
Terminated successfully

Execution 3
Processors: 4, Dimension: 3000x3000, Loops: 100
time elapsed: 0.593714
Terminated successfully
```



MPI vs MPI/OpenMP vs CUDA

Ακολουθεί διάγραμμα χρόνων εκτέλεσης ανάμεσα στις 3 υλοποιήσεις. Για τα MPI-MPI/OpenMP λήφθηκαν υπόψη οι χρόνοι εκτέλεσης για 9 processes.





Παρατηρήσεις:

- Ο λόγος που το MPI και το MPI/OpenMP παρουσιάζουν παρεμφερείς χρόνους για διαστάσεις 3000x3000 στο δεύτερο διάγραμμα είναι ότι έχουμε στη διάθεση μας 7 μηχανήματα με 2 πυρήνες έκαστως. Επομένως για την εκτέλεση με 9 processes και 4 threads κάποια processes θα τρέχουν στο ίδιο μηχάνημα, με αποτέλεσμα να μην εκτελούνται πλήρως παράλληλα αυτά τα processes.
- Στις διαστάσεις 30x30 ο χρόνος του hybrid είναι μικρότερος από του CUDA, διότι το πλήθος των υπολογισμών είναι μικρό αλλά υπάρχει μεγαλύτερο κόστος αντιγραφής του πίνακα από τη μνήμη της CPU στη μνήμη της GPU.