Game of Life

Κωστοπούλου Καλλιόπη - sdi1200084
Σπάχου Ευαγγελία - sdi1200169

ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ
ΣΕΠΤΕΜΒΡΙΟΣ 2017

ΠΕΡΙΕΧΟΜΕΝΑ

ΓΕΝΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ

Το παραδοτέο μας απαρτίζεται από τρεις φακέλους κάθε ένας απ' τους οποίους περιέχει την υλοποίηση του προγράμματος Game of Life που καταδεικνύει το όνομά του (MPI, MPI+OpenMP, Cuda).

Ο κώδικας των τριών διαφορετικών υλοποιήσεων βρίσκεται μέσα σε αυτούς τους φακέλους ενώ έξω από αυτούς βρίσκεται το report μας κι όποιο άλλο αρχείο μπορεί να κρίνουμε απαραίτητο.

Στο report αυτό παρουσιάζονται τόσο οι σχεδιαστικές επιλογές/παραδοχές που κάναμε κατα την υλοποίηση της άσκησης όσο και η πλήρης μελέτη της απόδοσης των προγραμμάτων.

Παραθέτουμε μια λίστα με όλες τις απαιτήσεις που υποχρεούμαστε να λάβουμε υπόψη μας κατα την υλοποίηση και το σχεδιασμό των προγραμμάτων όπως αυτές συζητήθηκαν εκτενώς στις διαλέξεις:

1) **Διαμοιρασμός δεδομένων σε Blocks**
2) **ISend / IReceive / IReduce (Non-blocking communication)**
3) **Datatypes**
4) **Αποφυγή Αντιγραφών**
5) **Χρήση των: ReceiveInit / SendInit + Start**
6) **Τοπολογίες – (για την ανακατανομή των διεργασιών και την ενδεχόμενη αύξηση της απόδοσης (reordering))**

Τα παραπάνω bullets λήφθηκαν όλα υπόψη μας και θα αναφερθούμε σε αυτά εκτενέστερα στην παράγραφο που αφορά τον σχεδιασμό/υλοποίηση του κώδικα.

ΕΝΤΟΛΕΣ ΜΕΤΑΓΛΩΤΤΙΣΗΣ / ΕΚΤΕΛΕΣΗΣ

Δεν κρίναμε απαραίτητη την δημιουργία αρχείου Makefile καθώς οι υλοποιήσεις μας δεν απαρτίζονται από πολλά αρχεία και η μεταγλώττιση επομένως του προγράμματος είναι μια απλή διαδικασία.

ΜΕΤΑΓΛΩΤΤΙΣΗ:

MPI: mpicc -o <output> -c mpi_GameOfLife.c funDefinitions.c -lm

MPI+OpenMP: mpicc -o <output> -c mpi_GameOfLife.c funDefinitions.c -lm

ΕΚΤΕΛΕΣΗ:

MPI: mpiexec –f machines –n <n> <output>

MPI+OpenMP: mpiexec –f machines –n <n> <output>

Για την απλοποίηση την υλοποίησης του προγράμματος (όπως προτάθηκε στις διαλέξεις και το eclass) κάνουμε τις εξής παραδοχές:

1. Απαιτούμε ο πίνακας (πλέγμα) να είναι τετράγωνος (NxN).
2. Απαιτούμε το πλήθος των workers στους οποίους θα διαμοιρασθεί ο παραπάνω πίνακας να είναι τετραγωνικός αριθμός. (x*x) και η ρίζα του (x) να διαιρεί τέλεια την κάθε πλευρά του πλέγματος (N % x = 0). Με αυτον τον τρόπο διασφαλίζουμε τον ισομοιρασμό του πίνακα σε διεργασίες.
   Από το σύνολικό αριθμό n των διεργασιών που δίνεται σαν όρισμα στην main (<n>) , οι n-1 διεργασίες θα είναι σύμφωνα με την υλοποίησή μας workers ενώ η μια υπολοιπόμενη διεργασία θα είναι ο Master.

Συνεπώς, οι επιτρεπόμενες γενικά τιμές που μπορεί να πάρει το n σύμφωνα με τον περιορισμό του πλήθους των διεργασιών είναι :

- 2 (1+1)
- 5 (4+1)
- 10 (9+1)
- 17 (16+1)
- 26 (25+1)
- 37 (36+1)
- 50 (49+1)

Ακόμα, ανάλογα το μέγεθος του προβλήματος ο παραπάνω αριθμός διεργασιών n που δίνεται μπορεί να μην είναι έγκυρος και συνεπώς τότε θα εκτυπώνεται μήνυμα λάθους.

ΣΧΕΔΙΑΣΜΟΣ/ΥΛΟΠΟΙΗΣΗ ΚΩΔΙΚΑ

MPI

Επιλέξαμε στην υλοποίησή μας να ξεχωρίσουμε τον ρόλο του master από αυτόν των workers. Για να το επιτύχουμε αυτό χρειάστηκε να δημιουργήσουμε έναν νέο communicator στο group του οποίου συμπεριλαμβάνονται μόνο οι διεργασίες workers (όχι ο master – 0).

Η MPI_Cart_create(..) παίρνοντας σαν όρισμα τον καινούριο αυτόν communicator δημιουργεί τοπολογία στην οποία «συμμετέχουν» και λαμβάνουν συντεταγμένες μόνο οι workers.

Οι workers πλέον επικοινωνούν μεταξύ τους μέσω του communicator που επιστρέφει η MPI_Cart_create(..). Η απεικόνιση των task_ids σε coordinates μπορεί να είναι αυθαίρετη και τα ids των διεργασιών του MPI_COMM_WORLD θεωρητικά μπορεί να μην είναι ίδια με αυτά του καινούριου communicator (reordering) (στην πράξη βέβαια δεν βλέπουμε να συμβαίνει κάτι τέτοιο), συνεπώς η πληροφορία που απαιτείται για την επικοινωνία μεταξύ των workers διατίθεται μόνο μέσω του καινούριου communicator.

Επικοινωνία μεταξύ των workers διενεργείται για την αποστολή γραμμών-στηλών-κελιών του υποπίνακα μιας διεργασίας σε άλλες διεργασίες που φιλοξενούν υποπίνακες γειτονικούς ως προς αυτόν (τον υποπίνακα).

Στο πρόγραμμα μας διενεργείται επικοινωνία μεταξύ των workers και του master για την αποστολή των αρχικών και τελικών υποπινάκων των πρώτων στον δεύτερο ο οποίος θα τους εκτυπώσει με τη σωστή σειρά στο αρχείο initial.txt και final.txt αντίστοιχα.

Κάθε worker αρχικοποιεί τον δικό του υποπίνακα κια γνωρίζει στο τέλος την κατάσταση του τελικού του υποπίνακα. Αυτούς τους δύο υποπίνακες λοιπόν, στέλνει στον master που θα αναλάβει την εκτύπωσή τους.

Η αποστολή από τους εργάτες προς τον master και η λήψη από την πλευρά του master γίνονται με non-blocking μεθόδους.

Για να γνωρίζει ο master πώς να εκτυπώσει τους ληφθέντες υποπίνακες (έστω τους αρχικούς) με τη σωστή σειρά πρέπει να λάβει πληροφορία και για τις συντεταγμένες της διεργασίας που αποστέλει τον καθ’ ένα απ’ αυτούς. Για να το επιτύχουμε αυτό ο κάθε worker στέλνει στον master μαζί με τον υποπίνακα του και τις συντεταγμένες του.

Δεδομένου ότι το μέγεθος του υποπίνακα είναι δυναμικό θεωρήσαμε πιο εύκολο να αποφύγουμε την δημιουργία datatype για αυτόν τον σκόπο και δεσμεύσαμε απλώς επιπλέον χώρο για τις συντεταγμένες (για 2 integers) στον ίδιο συνεχόμενο χώρο μνήμης όπου αποθηκεύται και ο υποπίνακας.
Εξαιτίας αυτού διενεργούνται τέσσερις συνολικά στο πρόγραμμα μας memcpy(..) για την αντιγραφή 4 συνολικά integers.
Προτιμήσαμε με άλλα λόγια να ρίξουμε αμελητέα την απόδοση κάνοντας αντιγραφή 16 συνολικά bytes προκειμένου να έχουμε πιο απλό κώδικα.

Κατα τ’ άλλα αποφεύγουμε οπουδήποτε αλλού τις περιττές αντιγραφές δημιουργώντας datatypes.

Για την αποστολή των στηλών ενός worker στον γειτονά του δημιουργείται datatype με τη βοήθεια της MPI_Type_create_subarray(..).

Για την αποστολή των γραμμών δεν θεωρήσαμε σκόπιμη τη χρήση datatype μιας και αυτές βρίσκονται ήδη σε συνεχόμενες θέσεις μνήμης.

Τέλος, για βέλτιστη απόδοση της παραλληλίας ο διαμοιρασμός του πλέγματος στις διεργασίες – workers όπως απαιτείτο, γίνεται σε blocks.

Για τις περισσότερες από τις παραπάνω αποφάσεις λήφθηκαν υπόψη μας οι οδηγίες/συμβουλές που δόθηκαν κατα τη διάρκεια των διαλέξεων.

MPI + OPEN_MP

Κάναμε μόνο «MPI only outside PARALLEL regions» και δεν επεκτεθήκαμε σε χρήση threads και για την επικοινωνία μεταξύ των γειτόνων.

Συγκεκριμένα προσθέσαμε την εντολή:

- **#pragma omp parallel for shared(fromGrid, toGrid) schedule(static, 1)**

στα for loops των δύο συναρτήσεων που υπολογίζουν τα εσωτερικά και εξωτερικά, αντίστοιχα, κελιά της επόμενης εκάστοτε γενιάς.

CUDA

Για “GENERATIONS” επαναλήψεις, εκτελείται η συνάρτηση nextGenerationCells(). Στη συνέχεια καλείται η συνάρτηση isDifferent() η οποία ελέγχει την προηγούμενη με την τωρινή μορφή του πίνακα και αν δεν έχει αλλάξει ή δεν υπάρχει περίπτωση να αλλάξει επιστρέφει 0 (false) και τερματίζει.

Το σώμα της συνάρτησης nextGenerationCells() περιέχει κατάλληλες αλλαγές (size_t x = blockIdx.x * blockIdDim.x + threadIdx.x;) για να τρέξει στο Device.

```
/* Update the grid STEP times */
for(timer=0; timer<GENERATIONS; timer++) {
   dim3 blockSize(blocks, blocks);
   dim3 dimGrid(width, heigt);
   nextGenerationCells<<<blockSize, dimGrid>>>(gpu_t, gpu_t1, width, height);
   /* Swap arrays */
   gpu_temp = gpu_t;
   gpu_t = gpu_t1;
   gpu_t1 = gpu_temp;
   if(!isDifferent(gpu_t, gpu_t1, height, width))
      break;
}
cudaMemcpy(t, gpu_t, width*height, cudaMemcpyDeviceToHost);
cudaMemcpy(t1, gpu_t1, width*height, cudaMemcpyDeviceToHost);
```

(Σημείωση: Λόγω τεχνικών προβλημάτων το πρόγραμμα Cuda δεν δοκιμάστηκε και δεν εκτελέστηκε. Ο κώδικας αναπτύχθηκε λοιπόν χωρίς έλεγχο και με βάση τη βιβλιογραφία.)

ΜΕΤΡΗΣΕΙΣ / ΜΕΛΕΤΗ ΚΛΙΜΑΚΩΣΗΣ

Έχοντας στη διάθεση μας 7 συνολικά μηχανήματα linux τρέξαμε αναγκαστικά το Game of Life με λιγότερες από 14 διεργασίες κάθε φορά.
Το γεγονός ότι διαχωρίσαμε τον master από τους workers μας αναγκάζει να χρησιμοποιούμε πάντα workers+1 διεργασίες, αλλά μιας και απαιτούμε το πλήθος των workers να είναι τετραγωνικός αριθμός και δεδομένου ότι το μέγιστο πλήθος διεργασιών που μπορούμε να δημιουργήσουμε είναι 14 δεν έχουμε καμία διαφορά στα run-times, speed-ups από αυτά που θα μπορούσαμε να έχουμε.
Θεωρούμε πως θα έχουμε μονάχα μια μικρή επιβάρυνση εξαιτίας αυτού στο efficiency.

Σημείωση: Στον υπολογισμό των δεδομένων στους παρακάτω πίνακες δεν λαμβάνουμε υπόψη μας τη μια διεργασία του master έτσι ώστε να θεωρήσουμε πως η εκτέλεση του προγράμματος με 2 εργασίες δηλαδή 2 workers είναι ακολουθιακό και να κάνουμε τις απαραίτητες συγκρίσεις.
Συνεπώς δεν αποτυπώνεται στον πίνακα των efficiency η παραπάνω αναφερθείσα επιβάρυνση.

Αριθμός Επανάληψης σε κάθε εκτέλεση (GENERATIONS) : 100

RUN-TIMES

MPI

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.981004	8.126680	31.483782	127.494491	501.533389	-
5	0.541111	2.101067	8.238805	32.021113	126.893522	-
10	0.272057	0.968613	3.722270	14.367514	57.055960	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

MPI+ OPEN_MP

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.205022	4.824189	19.377050	77.553762	311.085566	-
5	0.347278	1.279990	4.899290	19.548631	77.911680	-
10	0.205028	0.672406	2.574164	10.250654	40.934864	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

CUDA

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2						-
5						-
10						-
17						-
26						-
37						-

SPEED_UPS

MPI

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.0	1.0	1.0	1.0	1.0	-
5	3.660994	3.867882	3.821402	3.981576	3.952396	-
10	7.281577	8.390017	8.458221	8.873803	8.790202	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

MPI+ OPEN_MP

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.0	1.0	1.0	1.0	1.0	-
5	3.469906	3.768927	3.955073	3.967222	3.992798	-
10	5.877353	7.174518	7.527512	7.565738	7.599526	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

CUDA

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2						-
5						-
10						-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

EFFICIENCY

MPI

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.0	1.0	1.0	1.0	1.0	-
5	0.932224	0.809064	0.939802	0.985978	0.976689	-
10	0.966971	0.915248	0.95535	0.995394	0.988099	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

MPI+ OPEN_MP

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2	1.0	1.0	1.0	1.0	1.0	-
5	0.867476	0.942232	0.988768	0.991806	0.998199	-
10	0.653039	0.797169	0.83639	0.840638	0.844392	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

CUDA

#Tasks	Order of Matrix					
	576	1152	2304	4608	9216	-
2						-
5						-
10						-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

Παραθέτουμε ενδεικτικά και τους χρόνους που παίρνουμε όταν στο MPI συμπεριλάβουμε ελέγχους τερματισμού η υλοποίηση των οποίων επιτυγχάνεται με χρήση της MPI_Reduce(..) και συνεπώς global επικοινωνίας.

Παρατηρούμε, σαφώς, πως τα run-times είναι χειρότερα σε αυτήν την περίπτωση.

MPI + MPI_REDUCE

#Tasks	Order of Matrix					
	576	**1152**	**2304**	**4608**	**9216**	-
2	2.011794	7.895010	31.731410	127.887640	509.081500	-
5	0.528945	2.038642	8.148630	32.294369	127.939552	-
10	0.288714	0.939484	3.705569	14.426619	57.485592	-
17	-	-	-	-	-	-
26	-	-	-	-	-	-
37	-	-	-	-	-	-

ΣΥΜΠΕΡΑΣΜΑΤΑ

RUN-TIMES

MPI: Παρατηρούμε ότι η βελτίωση των χρόνων εκτέλεσης καθώς αυξάνεται το πλήθος των διεργασιών μέχρι και τους 9 workers είναι αξιοσημείωτη. Η διαφορά σαφώς, των χρόνων εκτέλεσης απο τον 1 worker στους 4 είναι πολύ μεγαλύτερη απ' ότι από τους 5 στους 9 καθώς αυξάνεται το κόστος της επικοινωνίας μεταξύ των διεργασιών.

MPI + MPI_REDUCE: Όπως ήταν αναμενόμεον η global επικοινωνία μεταξύ των διεργασιών μειώνει τους χρόνους εκτέλεσης του προγράμματος σε σχέση με αυτούς χωρίς MPI_Reduce.

OPENMP: Εδώ παρατηρούμε τη σημαντική βελτίωση των χρόνων εκτέλεσης όταν προστίθεται η δυνατότητα ύπαρξης παράλληλων threads για την εκτέλεση των for loops. Το πρόγραμμα MPI εκτελείται τώρα πολύ ταχύτερα και για τις αναλογίες των υπολογισμών ισχύουν ξανά τα παραπάνω εξαιτίας του MPI.

SPEED-UPS:

MPI: Παρατηρούμε, όπως περιμέναμε, ότι η επιτάχυνση στο απλό MPI είναι σημαντική τόσο από τον 1 worker στους 4 όσο κι από τους 4 στους 9. Προφανώς αν συνεχίζαμε να αυξάνουμε τις διεργασίες η αύξηση της επιτάχυνσης θα μειωνόταν σημαντικά.
Παρατηρούμε επίσης πως μεταξύ των διαφορετικών μεγεθών του πίνακα για δεδομένο πλήθος διεργασιών η επιτάχυνση μένει σχετικά σταθερή, γεγονός που δηλώνει την σταθερή μείωση του χρόνου εκτέλεσης κατα παράγοντα N σε όλες τις περιπτώσεις μεγεθών σε σχέση με την εκτέλεση με το προηγούμενο πλήθος διεργασιών.
Δεν δοκιμάσαμε, ωστόσο ακόμα μεγαλύτερα μεγέθη πίνακα.

OPENMP: Εδώ παρατηρούμε άλλοτε μια ελάχιστη μείωση κι άλλοτε αύξηση της επιτάχυνσης σε σχέση με αυτές που παίρναμε στο απλό MPI. Ο λόγος για τη μείωση είναι ενδεχομένως η ήδη σημαντική βελτίωση του χρόνου εκτέλεσης απο το openMP στο «ακολουθιακό» πρόγραμμα. Από εδώ βγάζουμε ίσως το συμπέρασμα ότι το MPI είναι πιο «σημαντικό» για το ακολουθιακό πρόγραμμα απ' ότι το MPI+OpenMP για το ακολουθιακό πρόγραμμα με OpenMp και σαφώς ότι τα for loops για τον υπολογισμό της επόμενης γενιάς έχουν αξιοσημείωτο overhead.

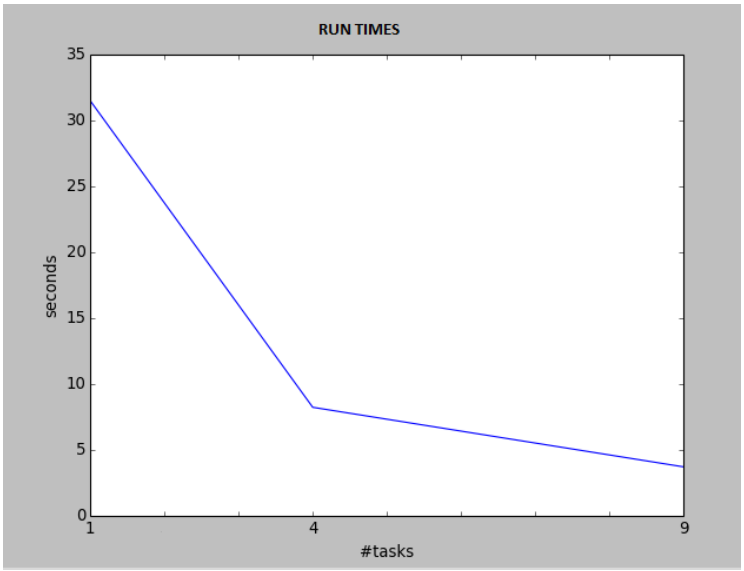
EFFICIENCY:

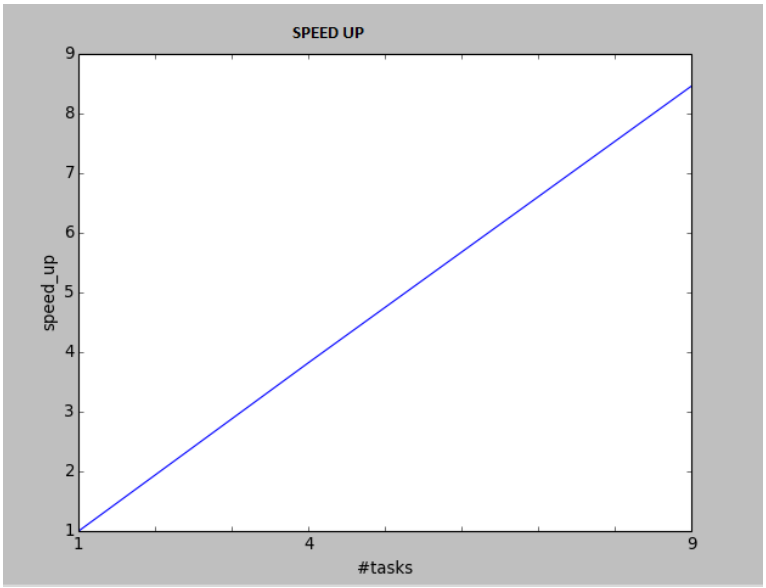
MPI: Η απόδοση όπως αναμέναμε για τα λίγα διαφορετικά πλήθη διεργασιών που χρησιμοποιήσαμε είναι πολύ μεγάλη. Φανταζόμαστε πως καθώς το πλήθος των διεργασιών αυξάνει κι άλλο η απόδοση θα πέφτει ακόμα κι αν οι χρόνοι εκτέλεσης μειώνονται..

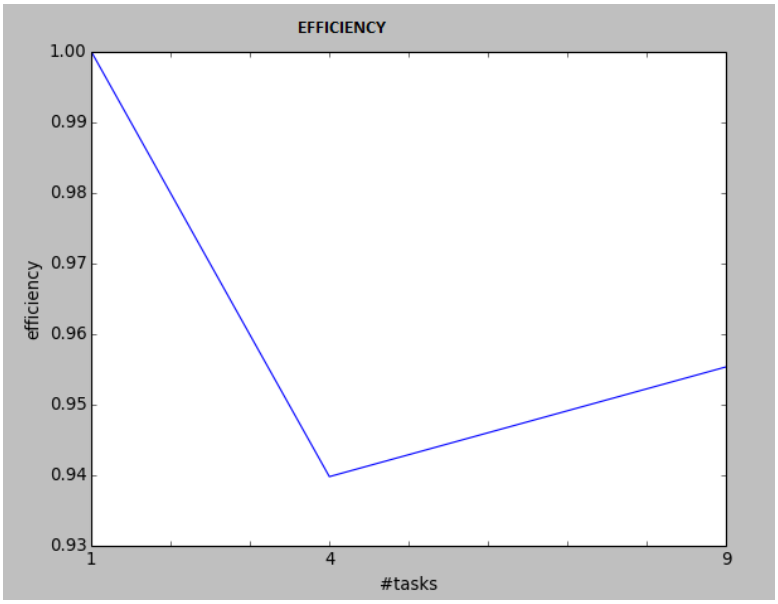
OPENMP: Δεδομένου ότι η επιτάχυνση από το MPI στο OpenMp όπως έπισημάνθηκε παραπάνω σε πολλές περιπτώσεις μειώνεται είναι προφανές και ότι το efficiency από το MPI στο OpenMP θα μειωθεί εξίσου.

ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ - MPI

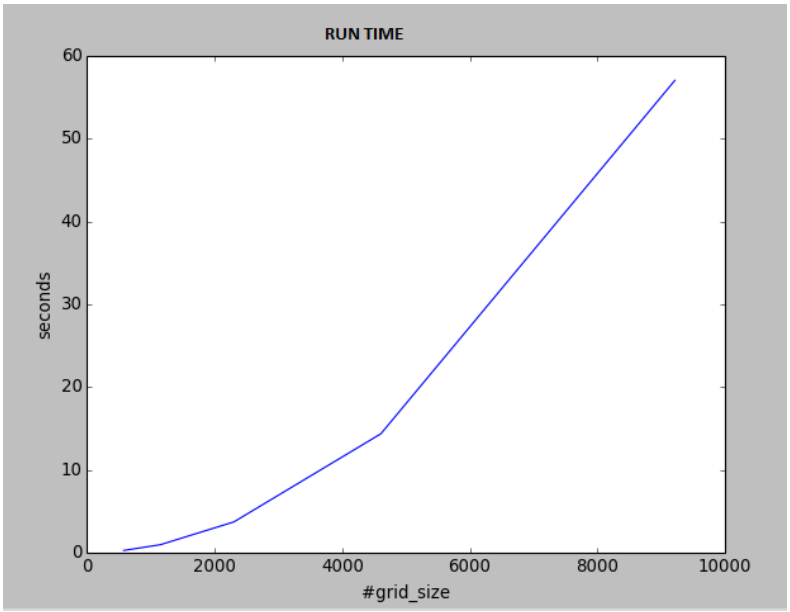
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ_2304

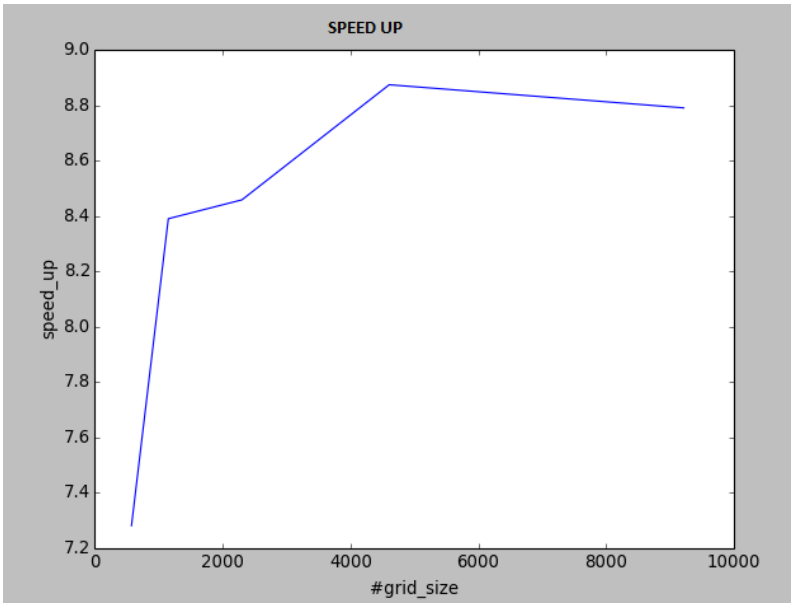


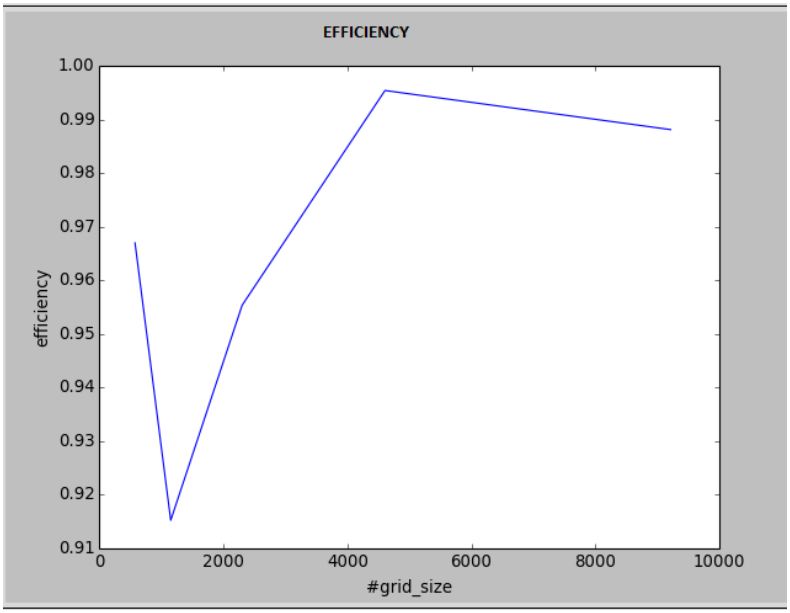




ΑΡΙΘΜΟΣ WORKERS : 9




SPEED UP
9.0
8.8
8.6
8.4
8.2
8.0
7.8
7.6
7.4
7.2
speed_up
0
2000
4000
6000
8000
10000
#grid_size

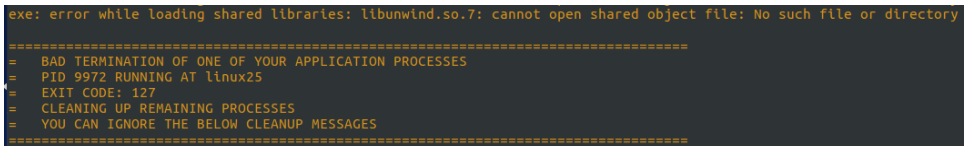

EFFICIENCY
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
efficiency
0
2000
4000
6000
8000
10000
#grid_size

ΣΗΜΕΙΩΣΕΙΣ:

Δυστυχώς λόγω των προβλημάτων που υπήρχαν με τα μηχανήματα της σχολής δεν μπορέσαμε να τελειοποιήσουμε την εργασία.

Τα προβλήματα που αντιμετωπίσαμε ήταν:

1. Cuda: γράψαμε τον κώδικα για την υλοποίηση αλλά δεν μπορέσαμε να να τον τρέξουμε έτσι ώστε να ολοκληρώσουμε ολες τις μετρησεις.
2. Paraver: για τις μετρήσεις απόδοσης και την οπτικοποίηση συμπεριφοράς ακολουθήσαμε τις οδηγίες που υπάρχουν στο σχετικό pdf αλλά στο σημείο της εκτέλεσης μας έβγαζε το ακόλουθο λάθος και δεν δημιουργούσε τα αρχεία που θα έπρεπε να δημιουργεί.



PS: Ελπίζουμε σε επιείκια σε ό,τι αφορά τα παραπάνω!!