9ο Εξάμηνο, Ακαδημαϊκό Έτος 2021-2022

Συστήματα Παράλληλης Επεξεργασίας

Αναφορά 1ης Εργαστηριακής Άσκησης: Εισαγωγή στην Παραλληλοποίηση σε Αρχιτεκτονικές Κοινής Μνήμης

parlab20

Ζάρα Στέλλα, 03117154
Λιάγκα Αικατερίνη, 03117208

Στην 1η εργαστηριακή άσκηση ζητήθηκε να τροποποιηθεί ο δοθέν σειριακός κώδικας για το Conway's Game of Life ώστε το πρόγραμμα να είναι πλέον παράλληλο και να ληφθούν και να σχολιαστουν μετρήσεις όταν το πρόγραμμα τρέχει για 1,2,4,6,8 πυρήνες διαδοχικά σε ταμπλό 64 × 64, 1024 × 1024 και 4096 × 4096 για 1000 γενιές.

Η μόνη αλλαγή που έπρεπε να γίνει στον κώδικα του Game_Of_Life.c που μας δόθηκε για να γίνει το πρόγραμμα παράλληλο ήταν να προστεθεί ανάμεσα στο πρώτο (μη παραλληλοποιήσιμο for του χρονου) και το δεύτερο for η παρακάτω γραμμή:

#pragma omp parallel for shared(N, current,previous) private(i,j,nbrs)

Με την παραπάνω εντολή, του OpenMP, ορίζουμε μια παράλληλη περιοχή και συγκεκριμένα ένα παράλληλο for loop. Ακόμα, ορίζουμε τις μεταβλητές N, current και previous ως shared, που σημαίνει ότι όλα τα threads έχουν πρόσβαση σε αυτές και μπορούν να γράψουν σε αυτές ή να τις διαβάσουν. Αντίστοιχα, οι μεταβλητές private είναι ιδιωτικές για κάθε thread και δεν μπορεί ένα thread να δει τι τιμή έχει ένα άλλο.

Για να δημιουργηθεί εκτελέσιμο για το πρόγραμμα δημιουργήθηκε το ακόλουθο Makefile

```
all: Game_Of_Life

Game_Of_Life: Game_Of_Life.c
	gcc -O3 -fopenmp -o Game_Of_Life Game_Of_Life.c

clean:
	rm Game_Of_Life
```

Στον scirouter όλες οι εκτελέσεις προγραμμάτων γίνονται με υποβολή της σχετικής εργασίας στο χρονοδρομολογητή parlab queue. Για αυτό και δημιουργήθηκαν τα αρχεία make_on_queue.sh για την εκτέλεση του make και τη δημιουργία του εκτελέσιμου και το run_on_queue.sh για την εκτέλεση του προγράμματος Game_Of_Life στα μηχανήματα του εργαστηρίου αφού υποβληθούν στην ουρά parlab queue.

Το make_on_queue.sh παρατίθεται παρακάτω

```
#!/bin/bash

##Descriptive name of job
#PBS -N make_Game_Of_Life

##Output and error files
#PBS -o make_Game_Of_Life.out
```

```
#PBS -e make_Game_Of_Life.err

##Machines
#PBS -l nodes=1:ppn=1

##Time
#PBS -l walltime=00:30:00

##Start
##Run make in src folder

module load openmp
cd /home/parallel/parlab20/a1
make
```

Με λίγα λόγια δεσμεύεται ένας πυρήνας ενός μηχανήματος για την δημιουργία του εκτελέσιμου Game_Of_Life, το αποτέλεσμα της εκτέλεσης αυτής (εκτέλεση του make του directory /home/parallel/parlab20/a1) αποθηκεύεται στο αρχειο make_Game_Of_Life.out ενώ αν προκύψει κάποιο error γράφεται στο αρχείο make_Game_Of_Life.err.

Το run_on_queue.sh παρατίθεται παρακάτω:

```
#!/bin/bash

##Describe job
#PBS -N run_Game_Of_Life

##Output and error files
#PBS -o run_Game_Of_Life.out
#PBS -e run_Game_Of_Life.err

##Machines
#PBS -l nodes=1:ppn=8

##Time
#PBS -l walltime=02:00:00

##Start
##Run make in src folder

core_num=(1 2 4 6 8)
```

```
for x in "${core_num[@]}"
do
        module load openmp
        cd /home/parallel/parlab20/a1
        export OMP_NUM_THREADS=${x}
        ./Game_Of_Life 64 1000
        ./Game_Of_Life 1024 1000
        ./Game_Of_Life 4096 1000
done
```

Τα ζητούμενα αποτελέσματα γράφονται στο αρχείο run_Game_Of_Life.out, ενώ αν προκύψει κάποιο error θα βρίσκεται στο αρχείο run_Game_Of_Life.err. Σε αυτήν την περίπτωση ζητείται να δεσμευτούν 8 πυρήνες ενός μηχανήματος (#PBS -l nodes=1:ppn=8) και με την βοήθεια του for loop εκτελείται το Game_Of_Life που έχουμε παράξει προηγουμένως διαδοχικα για 1,2,4,6,8 πυρήνες. Αναλυτικότερα η μεταβλητή OMP_NUM_THREADS περιγράφει τον αριθμό των νημάτων (πυρήνων) που επιτρέπεται να χρησιμοποιηθούν στην παράλληλη περιοχή οπότε για OMP_NUM_THREADS=2 θα χρησιμοποιηθούν 2 πυρήνες παρόλο που έχουμε δεσμεύσει 8.

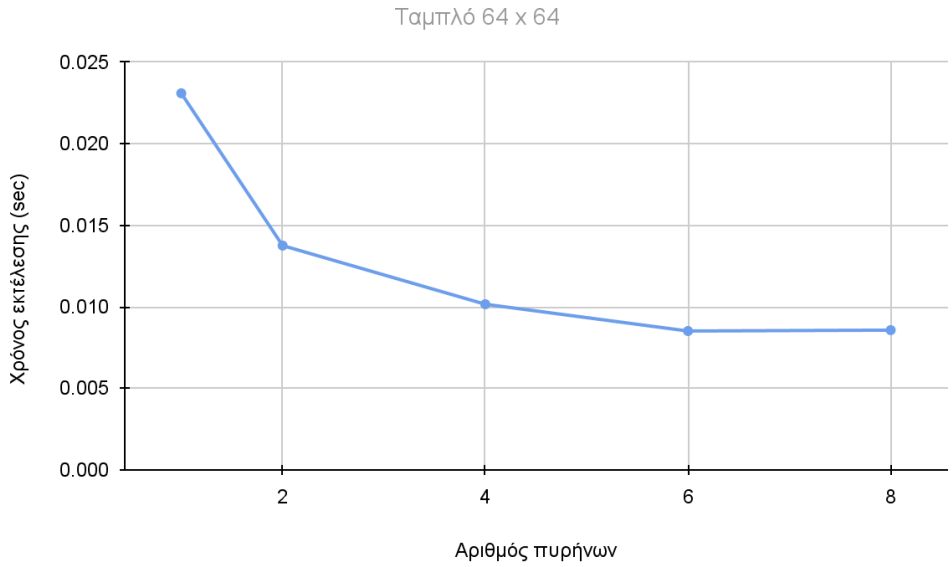
Τα αποτελέσματα του run_Game_Of_Life.out είναι οι χρόνοι εκτέλεσης για τους διάφορους αριθμούς πυρήνων και μεγεθών ταμπλό. Τα αποτελέσματα συνοψίζονται στον παρακάτω πίνακα:

	1	2	4	6	8
64x64	0.023096	0.013756	0.010158	0.008517	0.008571
1024x1024	10.973696	5.584990	2.780450	1.022132	0.771912
4096x4096	176.006599	89.968170	45.386430	29.384327	29.284225

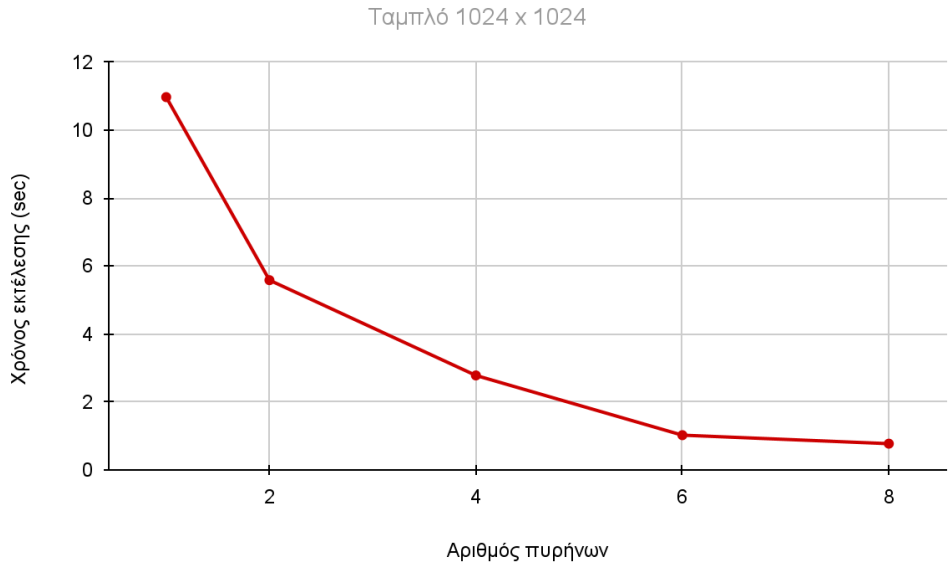
Πίνακας 1: Συνοπτικά τα αποτελέσματα της εκτέλεσης για διαφορετικά μεγέθη πίνακα και διαφορετικό αριθμό πυρήνων

Στη συνέχεια, για διευκόλυνση της ερμηνείας των αποτελεσμάτων, τα παραπάνω δεδομένα οπτικοποιήθηκαν με τη χρήση διαγραμμάτων και μελετήθηκε η μεταβολή του χρόνου εκτέλεσης σε σχέση με τον αριθμό των πυρήνων για κάθε μέγεθος πίνακα ξεχωριστά.

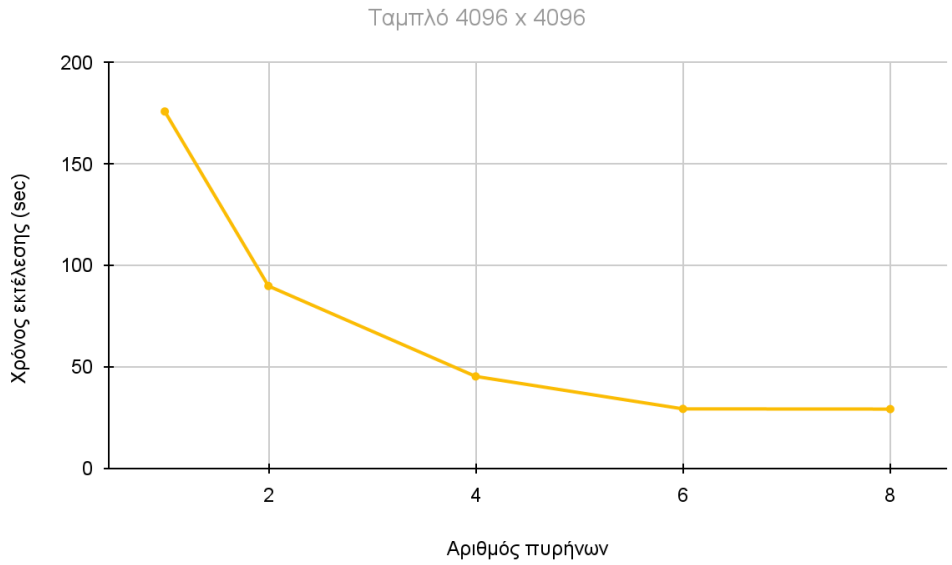
Τα διαγράμματα αυτά φαίνονται παρακάτω:



Διάγραμμα 1: Μεταβολή του χρόνου εκτέλεσης σε σχέση με τον αριθμό των πυρήνων για πίνακα μεγέθους 64x64

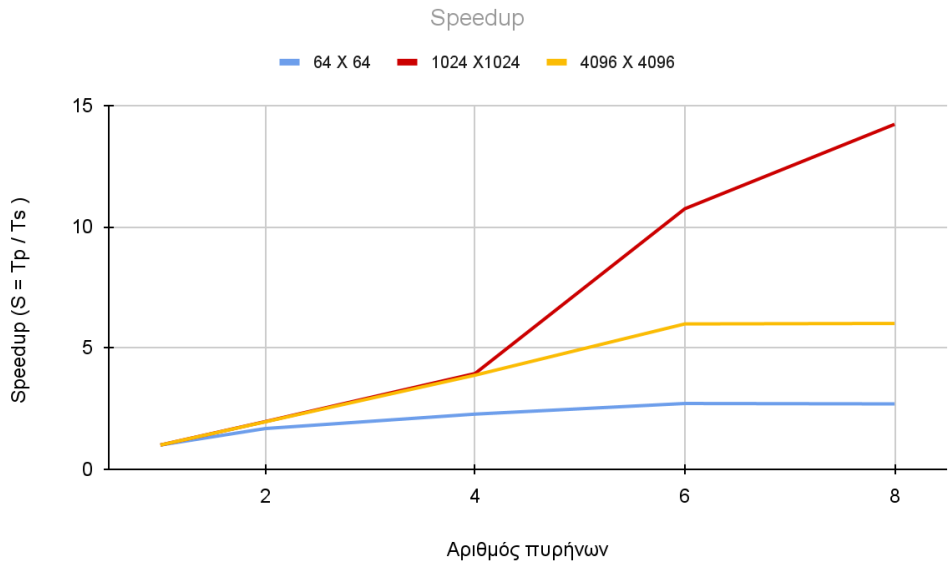


Διάγραμμα 2: Μεταβολή του χρόνου εκτέλεσης σε σχέση με τον αριθμό των πυρήνων για πίνακα μεγέθους 1024x1024

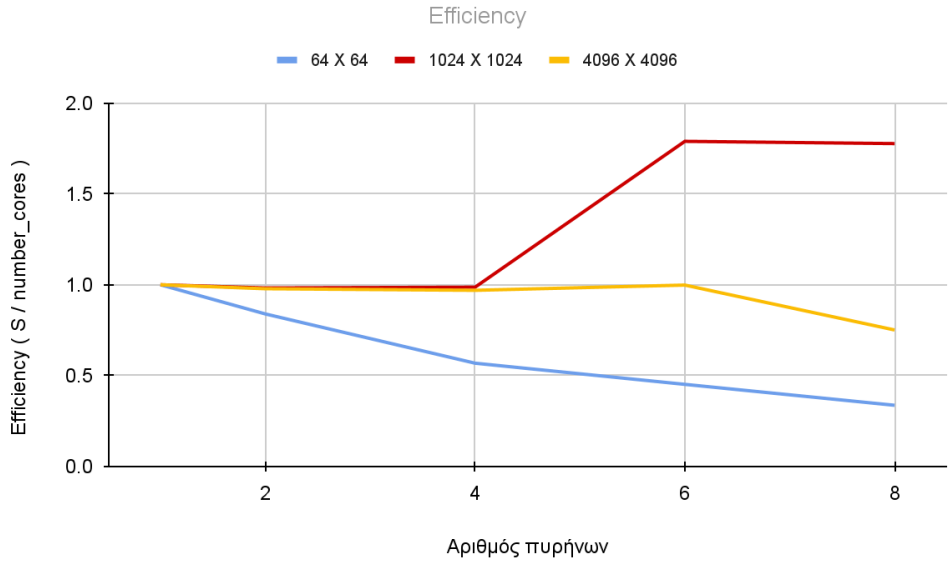


Διάγραμμα 3: Μεταβολή του χρόνου εκτέλεσης σε σχέση με τον αριθμό των πυρήνων για πίνακα μεγέθους 4096x4096

Επιπλέον για την αξιολόγηση των αποτελεσμάτων υπολογίστηκαν η επιτάχυνση (speedup) που δείχνει πόσες φορές πιο γρήγορο είναι το παράλληλο πρόγραμμα και η αποδοτικότητα (efficiency) που υποδεικνύει πόσο επιτυχημένη είναι η παραλληλοποίηση.



Διάγραμμα 4: Σύγκριση μεταβολής speedup ανάλογα με τον αριθμό των πυρήνων για τα 3 μεγέθη πίνακα



Διάγραμμα 5: Σύγκριση μεταβολής efficiency ανάλογα με τον αριθμό των πυρήνων για τα 3 μεγέθη πίνακα

Παρατηρείται ότι για μικρό μέγεθος ταμπλό (64X64) η παραλληλοποίηση δεν είναι ιδιαίτερα ωφέλιμη. Αναλυτικότερα, αν και οι χρόνοι εκτέλεσης μειώνονται, ο χρόνος που κερδίζεται με την αύξηση πυρήνων (threads) είναι πολύ μικρός με αποτέλεσμα η επιτάχυνση να είναι σχεδόν αμελητέα, ενώ η αποδοτικότητα, δηλαδή το ποσοστό του χρόνου που κάθε επεξεργαστής κάνει χρήσιμη δουλειά φαίνεται να μειώνεται όσο αυξάνονται οι πυρήνες. Όλα τα παραπάνω είναι αναμενόμενα μιας και γνωρίζουμε ότι αν στο παραλληλοποιημένο πρόγραμμα κάθε thread εκτελεί πολύ μικρό όγκο εργασιών τότε το overhead της αρχικοποίησης και τερματισμού των threads υπερβαίνει την χρήσιμη δουλειά που εκτελούν.

Είναι φανερό ότι για μεγαλύτερα μεγέθη πίνακα η παραλληλοποίηση είναι πιο ωφέλιμη, αφού παρατηρουνται σημαντικά μεγαλύτερες επιταχύνσεις και η αποδοτικότητα αυξάνεται για μεγαλύτερο αριθμό threads.

Βέβαια, φαίνεται ότι για μεγάλο αριθμό threads οι χρόνοι εκτέλεσης σταθεροποιούνται ή παρατηρείται ελάχιστη αύξηση. Επίσης σταθεροποιούνται (ή παρατηρείται ακόμα και μικρή μείωση) τα speedup και τα efficiency. Αυτό ενδεχομένως οφείλεται στο overhead που προκύπτει επειδή τα threads μοιράζονται πεπερασμένους πόρους. Ένα παράδειγμα για το πως αυτό μπορεί να επηρεάσει αρνητικά τον τελικό χρόνο εκτέλεσης είναι ότι ταυτόχρονα πολλά threads προσπαθουν να έχουν πρόσβαση σε δεδομένα με αποτέλεσμα η κοινή μνήμη cache να ανανεώνεται συνεχώς επομένως να πρέπει το ίδιο thread να κάνει εν τέλει περισσότερα fetch από την μνήμη.