ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συστήματα Παράλληλης Επεξεργασίας

Άσκηση 1η

Ιωάννης Ρέκκας 03119049

Αναστάσιος Στέφανος Αναγνώστου 03119051

Γεώργιος Αναστασίου 03119112

12 Νοεμβρίου 2023

Περιεχόμενα

1 Συλλογή Δεδομένων

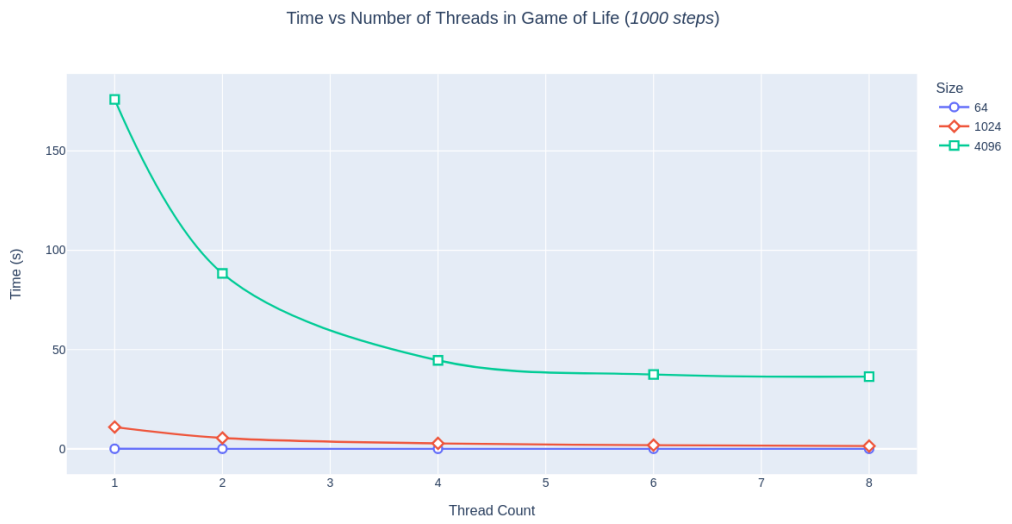
Αρχικά, το πρόγραμμα παραλληλοποιήθηκε χρησιμοποιώντας την κατάλληλη μακροεντολή του OpenMP, όπως φαίνεται στον κώδικα ;; (βλ. Παράρτημα). Για την συλλογή δεδομένων δεσμεύτηκε ένας υπολογιστικός κόμβος με οκτώ πυρήνες και εκτελέστηκε το script ;;. Το εκτελέσιμο δημιουργήθηκε με την μεταγλώττιση όπως φαίνεται στο Makefile ;;. Τα δεδομένα εξόδου φαίνονται παρακάτω.

```
Thread Count = 1
GameOfLife: Size 64 Steps 1000 Time 0.023253
GameOfLife: Size 1024 Steps 1000 Time 10.968962
GameOfLife: Size 4096 Steps 1000 Time 175.976354
Thread Count = 2
GameOfLife: Size 64 Steps 1000 Time 0.013578
GameOfLife: Size 1024 Steps 1000 Time 5.457094
GameOfLife: Size 4096 Steps 1000 Time 88.317703
Thread Count = 4
GameOfLife: Size 64 Steps 1000 Time 0.010049
GameOfLife: Size 1024 Steps 1000 Time 2.723685
GameOfLife: Size 4096 Steps 1000 Time 44.545233
Thread Count = 6
GameOfLife: Size 64 Steps 1000 Time 0.009147
GameOfLife: Size 1024 Steps 1000 Time 1.833331
GameOfLife: Size 4096 Steps 1000 Time 37.423192
Thread Count = 8
GameOfLife: Size 64 Steps 1000 Time 0.009746
GameOfLife: Size 1024 Steps 1000 Time 1.376557
GameOfLife: Size 4096 Steps 1000 Time 36.365997
```

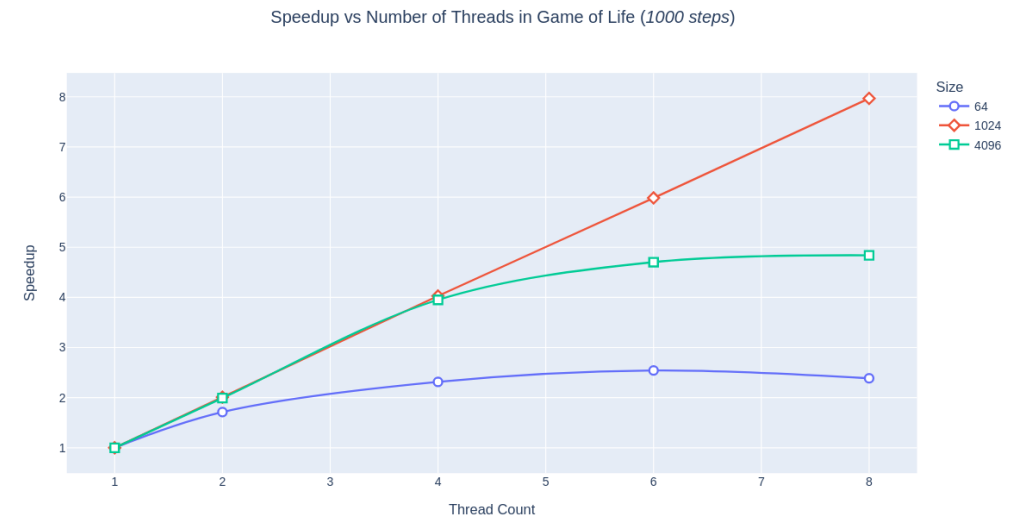
Σχήμα 1: Δεδομένα Εξόδου Game Of Life

2 Σύγκριση Μετρήσεων

Οι μετρήσεις συνοψίζονται στα παρακάτω γραφήματα.



(α′) Χρόνος εκτέλεσης του Game Of Life ως προς το πλήθος των διεργασιών για διάφορα μεγέθη του παιχνιδιού.



(β′) Επιτάχυνση του Game Of Life για διάφορα μεγέθη του παιχνιδιού.

Στο γράφημα ;; απεικονίζονται τρεις καμπύλες, μία για κάθε μέγεθος της προσομοίωσης Game of Life, οι οποίες αποτυπώνουν τον χρόνο εκτέλεσης της προσομοίωσης για διάφορα πλήθη νημάτων. Στο γράφημα ;; απεικονίζονται τρεις καμπύλες, οι οποίες αποτυπώνουν την επιτάχυνση (speedup) για καθένα από τα μεγέθη της προσομοίωσης.

3 Ερμηνεία Αποτελεσμάτων

Αρχικά, φαίνεται ότι τόσο το 64 όσο και το 4096 δεν κλιμακώνουν.

Στο μέγεθος προσομοίωσης 64×64 η εκτέλεση είναι τόσο γρήγορη ώστε η δημιουργία νημάτων και μόνο να προκαλεί τόση καθυστέρηση ώστε το πρόγραμμα να μην κλιμακώνει. Ήδη από τα 2 νήματα και πάνω η κλιμάκωση δεν είναι ιδανική. Σημειώνεται επιτάχυνση, απλώς όχι τόση όση αναμένεται από τον παραλληλισμό.

Στο μέγεθος προσομοίωσης 4096×4096 η εκτέλεση είναι χρονοβόρα, άρα η δημιουργία νημάτων δεν προκαλεί συγκρίσιμη καθυστέρηση. Προκαλείται, όμως, συμφόρηση στον δίαυλο επειδή πλέον δεν χωράνε όλα τα δεδομένα στη cache και έτσι τα νήματα αιτούνται δεδομένα συχνότερα από την κύρια μνήμη. Η ύπαρξη του διαύλου σειριοποιεί ένα κομμάτι του προγράμματος αναγκάζοντας κάποια νήματα να περιμένουν άλλα για να πάρουν τα δικά τους δεδομένα. Αυτή η συμπεριφορά βέβαια εκδηλώνεται στην δημιουργία 6 και περισσότερων νημάτων. Μέχρι και τα 4 νήματα η κλιμάκωση είναι σχεδόν ιδανική και ο δίαυλος μπορεί να ανταπεξέλθει στις αιτήσεις.

Αντίθετα στο μέγεθος 1024×1024 παρατηρούμε ιδανικό γραμμικό speedup για όλα τα πλήθη νημάτων. Εκεί οι παράμετροι εναρμονίζονται έτσι ώστε να μην υπάρχει συμφόρηση στον διαυλο και η κατανομή εργασίας στα νήματα είναι τέλεια.

4 Παράρτημα

Listing 1: Parallelized Game Of Life

```
...
for ( t = 0 ; t < T ; t++ ) {
    #pragma omp parallel for private(nbrs, i, j) shared(previous, current)
    for ( i = 1 ; i < N-1 ; i++ )
        for ( j = 1 ; j < N-1 ; j++ ) {
            nbrs = previous[i+1][j+1] + previous[i+1][j] + previous[i+1][j-1] \
                 + previous[i][j-1] + previous[i][j+1] \
                 + previous[i-1][j-1] + previous[i-1][j] + previous[i-1][j+1];
            if ( nbrs == 3 || ( previous[i][j]+nbrs ==3 ) )
                current[i][j]=1;
            else
                current[i][j]=0;
        }
        #ifdef OUTPUT
        print_to_pgm(current, N, t+1);
        #endif
        //Swap current array with previous array
        swap=current;
        current=previous;
        previous=swap;
}
...
```

Listing 2: Bash Script to build Game Of Life

```
#!/bin/bash

## Give the Job a descriptive name
#PBS -N make_omp_gol

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

## How many machines should we get?
#PBS -l nodes=1:ppn=1

##How long should the job run for?
#PBS -l walltime=00:01:00

## Start
## Run make in the src folder (modify properly)

module load openmp
cd /home/parallel/parlab30/a1
make
```

Listing 3: 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
```

Listing 4: Bash Script to Gather Measurements

```
#!/bin/bash

## Give the Job a descriptive name
#PBS -N run_gol

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

## How many machines should we get?
#PBS -l nodes=1:ppn=8

##How long should the job run for?
#PBS -l walltime=00:10:00

## Start
## Run make in the src folder (modify properly)

module load openmp
cd /home/parallel/parlab30/a1

threads=(1 2 4 6 8)
sizes=(64 1024 4096)
speed=1000

for thread in "${threads[@]}"
do
    export OMP_NUM_THREADS="$thread"
    for size in "${sizes[@]}"
    do
        ./Game_Of_Life "$size" "$speed"
    done
done
```