Εργασία 2: MPI

Πρώτη έκδοση

Κωνσταντίνος Σαμαράς-Τσακίρης

6/1/2015

Στόχος

Η υλοποίηση ενός διανεμημένου αλγορίθμου kNN με χρήση MPI.

Σχόλια

- Εκτός από την παρεχόμενη έκδοση, ο κώδικας είναι διαθέσιμος στο Γιτηυβ: https://github.com/Oblynx/parallel course-projects/tree/61e8e2609b0b78e9adfc2d7f44e34cddb49ae1d1/proj2
- Χρησιμοποιείται C++11 και MPI-3
- Εκ των υστέρων διαπιστώνω ότι το Hellasgrid δεν υποστηρίζει MPI-3, οπότε θα χρειαστεί μια μικρή αλλαγή για να τρέξει εκεί

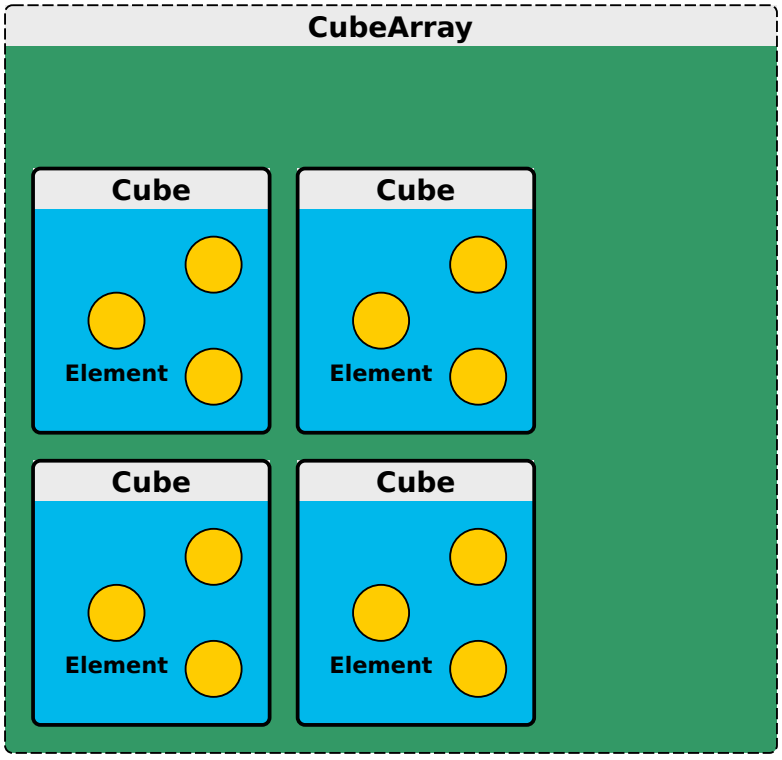
1 Ανάλυση του αλγορίθμου

Ο αλγόριθμος χωρίζει το χώρο σε κύβους (βλ. διάγραμμα 1). Αν δεχθούμε ότι δεν υπάρχουν κενοί κύβοι, οι βασικοί υποψήφιοι S κάθε ερωτήματος Q είναι το σύνολο C_{Q1} των σημείων που περιλαμβάνει ο κύβος που περιέχει το Q και όλοι οι γείτονές του. Αν δεχθούμε ότι ενδεχομένως υπάρχουν κενοί κύβοι, τότε με δεδομένη τη χωρική κατανομή των C υπάρχει συγκεκριμένη πιθανότητα το C_{Q1} να είναι το σύνολο των βασικών υποψηφίων και η πιθανότητα αυτή αυξάνεται ραγδαία κάθε φορά που επεκτείνεται το $C_{Q\,i} \longrightarrow C_{Q\,(i+1)}$, προσθέτοντας όλους τους γειτονικούς του κύβους. Αυτή η σκέψη επιτρέπει ακόμη λεπτότερο καταμερισμό του χώρου.

Το μέγεθος του προβλήματος μπορεί να επιβάλλει το χωρισμό σε πολλές διεργασίες. Για να επιτευχθεί αυτό, κάθε διεργασία είναι υπεύθυνη για ένα κυβικό τμήμα του χώρου (που αποτελείται από πολλούς από τους προηγούμενους κύβους) και γνωρίζει μόνο τα σημεία C, Q που ανήκουν σε αυτό. Με αυτό το χωρισμό του προβλήματος όμως, αν ζητηθεί Q στα όρια του χώρου ευθύνης μιας διεργασίας θα απαιτηθεί γνώση των στοιχείων C που βρίσκονται στους γειτονικούς κύβους άλλων διεργασιών.

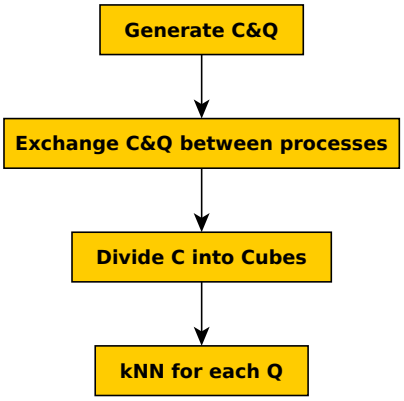
Πρώτη σκέψη για την επίλυση αυτού του προβλήματος είναι η ανταλλαγή των απαραίτητων πληροφοριών μεταξύ των διεργασιών, όταν παρίσταται τέτοια ανάγκη. Το κόστος των επικοινωνιών όμως είναι μεγάλο. Σε δεύτερη σκέψη οι επικοινωνίες μπορούν να αποφευχθούν, αν θεωρήσουμε αμελητέα την πιθανότητα το S να εκτείνεται σε χώρο μεγαλύτερο από $C_{Q\,m}$ – αν μάλιστα υποθέσουμε ότι δεν υπάρχουν κενοί κύβοι, τότε $m = 1$. Σε αυτήν την περίπτωση, σε στάδιο των αρχικών επικοινωνιών για το διαμοιρασμό των σημείων μπορούμε να στείλουμε τα σημεία που βρίσκονται στο σύνορο του χώρου 2 διεργασιών και στις 2 συνορεύουσες διεργασίες, όχι μόνο σε αυτήν που πραγματικά της ανήκουν[1]. Η διαδικασία αυτή ονομάζεται overlap και γίνεται σε βάθος m κύβων από το σύνορο.

[1] Αν το σημείο βρίσκεται σε γωνία συνόρου μπορεί να μην ανήκει μόνο σε 2, αλλά σε 3 ή και 8 διεργασίες.



Σχήμα 1: Οργάνωση καταμερισμού του χώρου – Ελεμεντ είναι ένα στοιχείο του συνόλου C και ὓβεΑρραψ είναι η περιοχή του χώρου που αντιστοιχεί σε κάθε διεργασία

Η πορεία του προγράμματος παρουσιάζεται στο διάγραμμα 2.



Σχήμα 2: Ροή προγράμματος

Επικοινωνία μεταξύ διεργασιών

Τόσο για τα σημεία C όσο και για τα Q η διαδικασία της επικοινωνίας είναι ακριβώς η ίδια:

1. Κάθε διεργασία δημιουργεί N/P τυχαία σημεία σε όλο το χώρο και υπολογίζει τις διεργασίες στις οποίες πρέπει να σταλούν.
2. Alltoall επικοινωνία του πλήθους των σημείων που θα αποσταλούν από κάθε διεργασία σε κάθε άλλη
3. Alltoallv επικοινωνία για την αποστολή των σημείων

Επειδή κάποιοι υπολογισμοί στα πλαίσια αυτής της διαδικασίας μπορούν να επικαλυφθούν με μεταφορές χρησιμοποιούνται nonblocking collective communications με τις συναρτήσεις MPI_Ialltoall και MPI_Ialltoallv που ορίζονται στο πρότυπο MPI-3. Για τη συλλογή μετρήσεων από το Hellasgrid, επειδή δεν υπάρχει MPI-3, αυτές θα αντικατασταθούν με τις αντίστοιχες blocking.

Επίλυση kNN

Ο αλγόριθμος για την επίλυση του προβλήματος kNN, με την τεχνική που περιγράφηκε παραπάνω, δε χρειάζεται επικοινωνία με άλλες διαδικασίες. Για κάθε ερώτημα Q εκτελεί την ακόλουθη απλή διαδικασία:

Αλγόριθμος 1 *query*()

1. Εύρεση κύβου *qloc* μέσα στα όρια του οποίου βρίσκεται το Q
2. Όρια αναζήτησης $searchLim = qloc$
3. $search(kNN)$
4. Αν δε βρέθηκαν σημεία ή η απόσταση του Q από το σύνορο του *qloc* είναι μικρότερη από την απόσταση του πιο απομακρυσμένου kNN
   α′) Διεύρυνση χώρου αναζήτησης (συμπερίληψη όλων των κύβων που συνορεύουν με τον τωρινό χώρο αναζήτησης)
   β′) $search(kNN)$
   γ′) Επανάλαβε όσο δεν έχουν βρεθεί k γείτονες

Αλγόριθμος 2 $search(kNN)$

1. Για κάθε κύβο *cube* στο χώρο αναζήτησης
   α′) Για κάθε στοιχείο *elt* στο *cube*
      i. Αν η απόσταση του *elt* από το Q είναι μικρότερη από την απόσταση του *top*, που είναι το πιο απομακρυσμένο kNN από το Q
         A′. $kNN- = top$
         B′. $kNN+ = elt$

Αν και η πιθανότητα να χρειαστεί επικοινωνία με άλλες διαδικασίες θεωρήθηκε μηδενική, σε περίπτωση που κάτι τέτοιο απαιτούνταν ο αλγόριθμος θα το αντιλαμβανόταν και θα αιτούνταν επικοινωνία. Σε αυτή την έκδοση όμως η αίτηση επικοινωνίας δεν έχει υλοποιηθεί και προκαλεί exception.

2 Έλεγχος ορθότητας

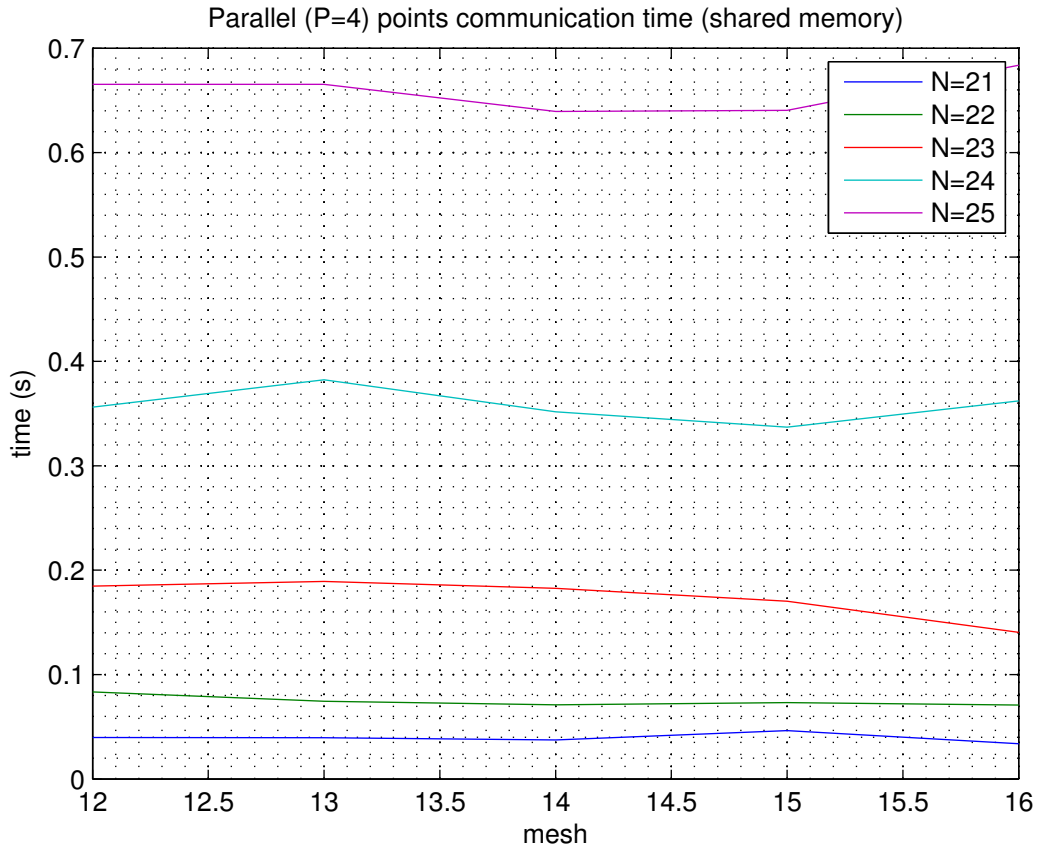
Σε αυτό το πρόβλημα δεν υπάρχει γρήγορος τρόπος ελέγχου ορθότητας της λύσης, σε αντίθεση με την προηγούμενη εργασία. Οπότε η ορθότητα του κώδικα τεκμηριώνεται από τη σωστή συμπεριφορά σε testcases με γνωστή λύση. Επειδή τα στάδια της επικοινωνίας και της επίλυσης kNN είναι ανεξάρτητα, ελέγχονται χωριστά. Ο έλεγχος της επικοινωνίας γίνεται στο αρχείο test_mpi_transfers.cpp, ενώ ο έλεγχος του αλγορίθμου γίνεται κυρίως σε σειριακή εκτέλεση στο αρχείο test_kNNsingle.cpp. Στην παράλληλη εκδοχή εξετάζεται αν λειτουργεί σωστά και το overlap. Για τον έλεγχο του kNN χρησιμοποιούνται 2 testcases:

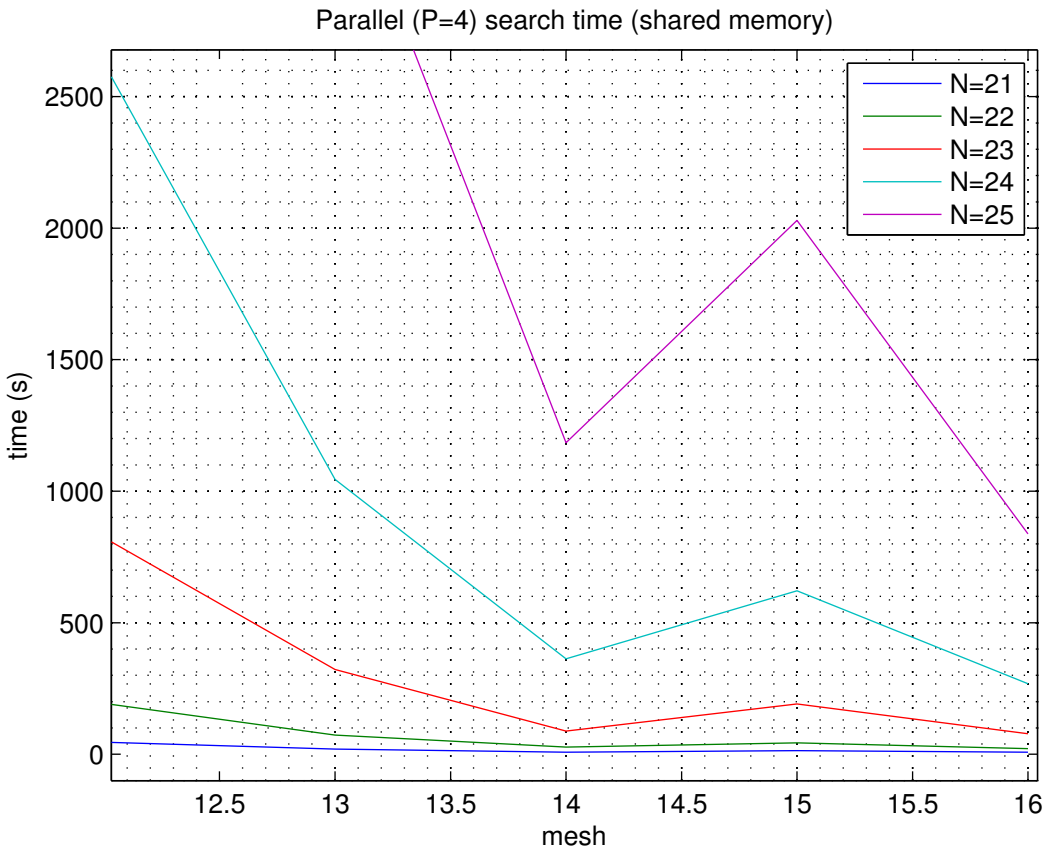
- Εισαγωγή πολλών τυχαίων και λίγων επιλεγμένων σημείων C στο χώρο. Επιλογή Q κοντά στα επιλεγμένα C. Μεγάλο k. Αναμενόμενο αποτέλεσμα: Στους kNN συμπεριλαμβάνονται τα επιλεγμένα σημεία, μαζί με κάποια από τα τυχαία που γειτονεύουν. Στοχευμένος έλεγχος και της συμπεριφοράς στα όρια του χώρου (overlap).
- Εισαγωγή C σε καθορισμένες θέσεις πλέγματος. Επιλογή Q. Αναμενόμενο αποτέλεσμα: Τα γειτονικά με το Q σημεία του πλέγματος, που είναι γνωστά.

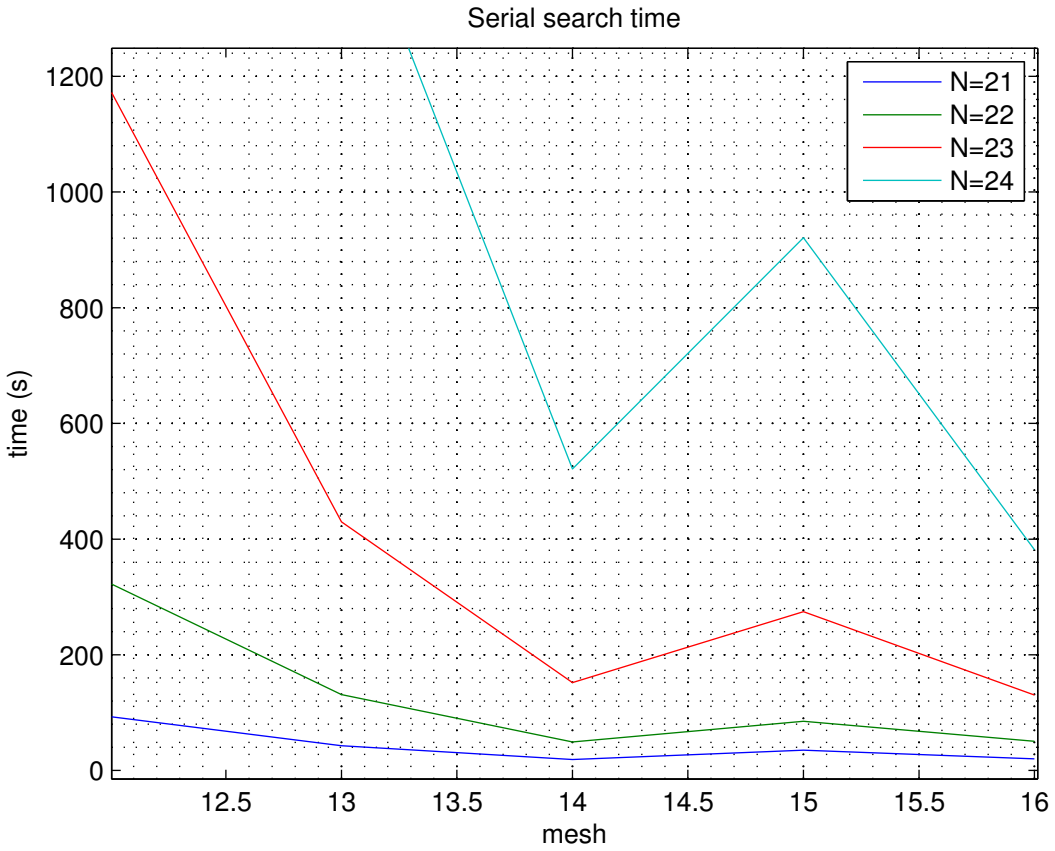
Για τον έλεγχο των επικοινωνιών επιβεβαιώνεται ότι τα σημεία που δημιουργήθηκαν από κάθε διαδικασία έφθασαν σε όλους τους αναμενόμενους προορισμούς τους.

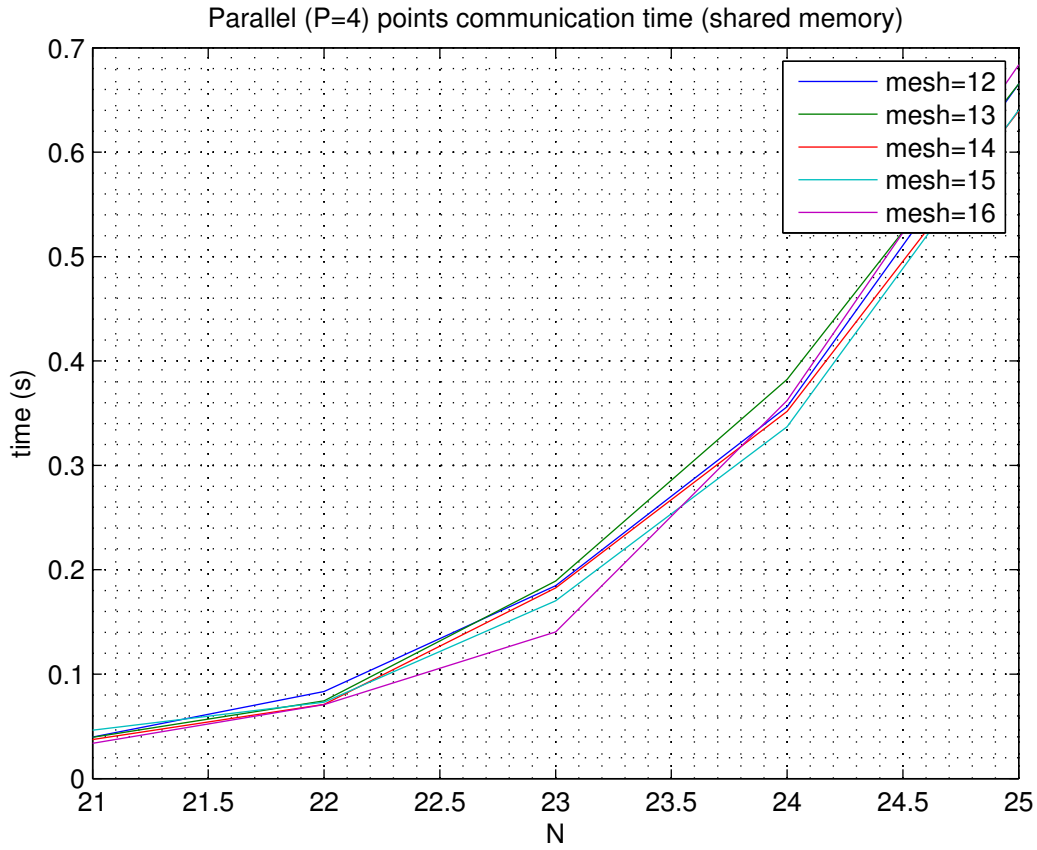
3 Αποτελέσματα

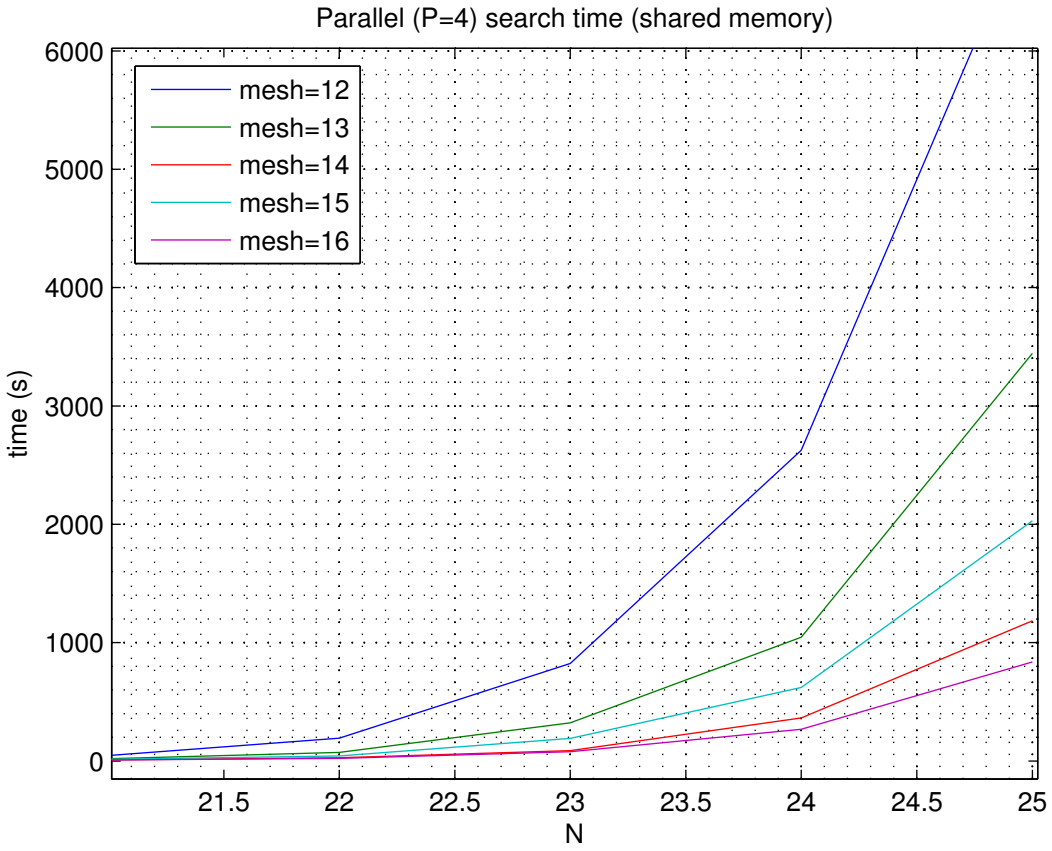
Λόγω αδυναμίας σύνδεσης με το Hellasgrid δεν έχουν ακόμη ληφθεί αποτελέσματα από εκεί. Θα παραδοθούν σε επόμενη έκδοση αυτής της εργασίας. Τώρα παρατίθενται μερικά αποτελέσματα από μελέτη σε έναν υπολογιστή με 4 διεργασίες. Όπως φαίνεται από τα $t - N$ γραφήματα παράλληλου και σειριακού χρόνου αναζήτησης, η μέγιστη επιτάχυνση των υπολογισμών με 4 διεργασίες σε σχέση με τη σειριακή περίπτωση είναι $< \times 2$.

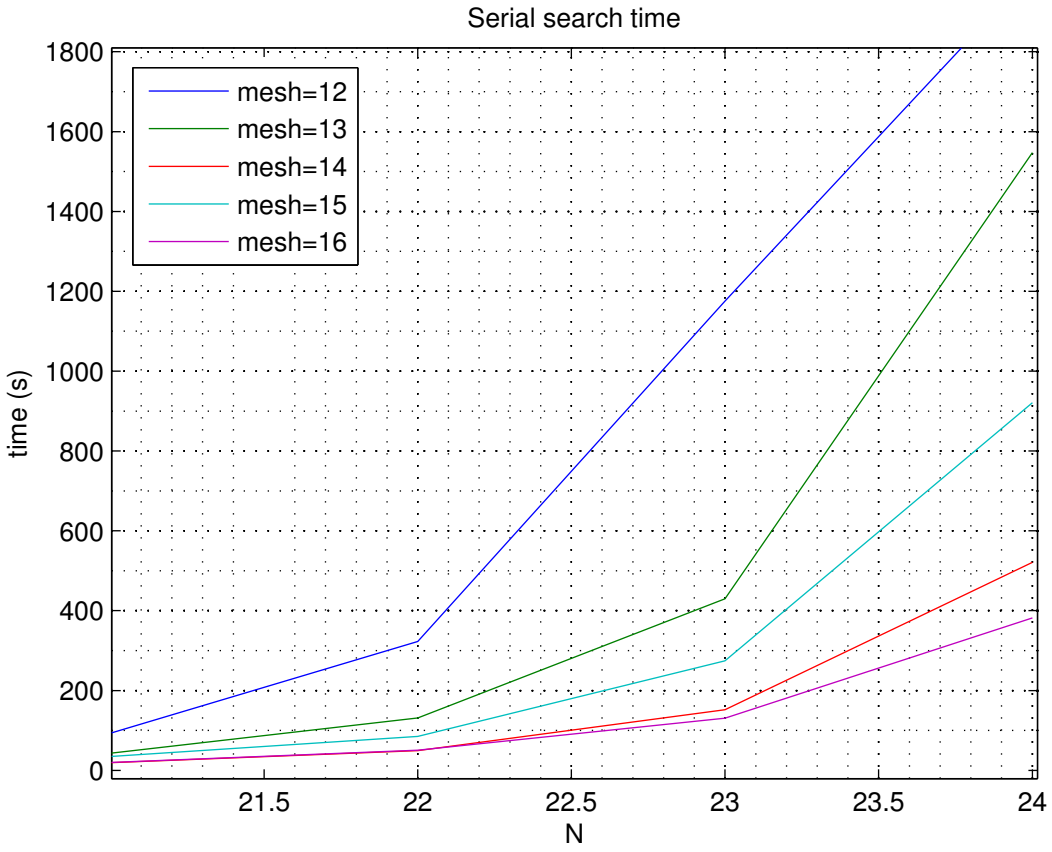

Parallel (P=4) points communication time (shared memory)
time (s)
mesh
N=21
N=22
N=23
N=24
N=25
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
12
12.5
13
13.5
14
14.5
15
15.5
16


Parallel (P=4) search time (shared memory)
time (s)
mesh
N=21
N=22
N=23
N=24
N=25
0
500
1000
1500
2000
2500
12.5
13
13.5
14
14.5
15
15.5
16


Serial search time
time (s)
mesh
N=21
N=22
N=23
N=24
0
200
400
600
800
1000
1200
12.5
13
13.5
14
14.5
15
15.5
16


Parallel (P=4) points communication time (shared memory)
time (s)
N
mesh=12
mesh=13
mesh=14
mesh=15
mesh=16
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
21
21.5
22
22.5
23
23.5
24
24.5
25


Parallel (P=4) search time (shared memory)
mesh=12
mesh=13
mesh=14
mesh=15
mesh=16
time (s)
N
0
1000
2000
3000
4000
5000
6000
21.5
22
22.5
23
23.5
24
24.5
25


Serial search time
mesh=12
mesh=13
mesh=14
mesh=15
mesh=16
time (s)
N
0
200
400
600
800
1000
1200
1400
1600
1800
21.5
22
22.5
23
23.5
24