Εργασία 2: MPI

Τελική έκδοση

Κωνσταντίνος Σαμαράς-Τσακίρης

10/1/2015

Στόχος

Η υλοποίηση ενός διανεμημένου αλγορίθμου kNN με χρήση MPI.

Σχόλια

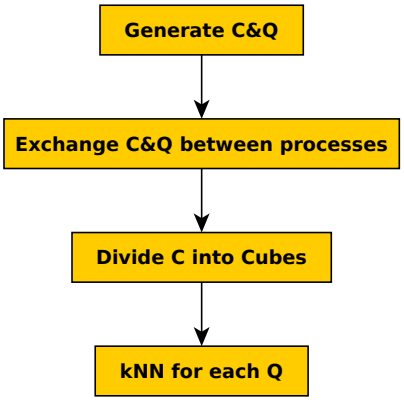
- Εκτός από την παρεχόμενη έκδοση, ο κώδικας είναι διαθέσιμος στο Github
  - https://github.com/Oblynx/parallel-course-projects/tree/master/proj2
  - Η έκδοση του κώδικα που χρησιμοποιήθηκε εδώ είναι το tag proj2_v1.2
- Χρησιμοποιείται C++11 και MPI-3

1 Ανάλυση του αλγορίθμου

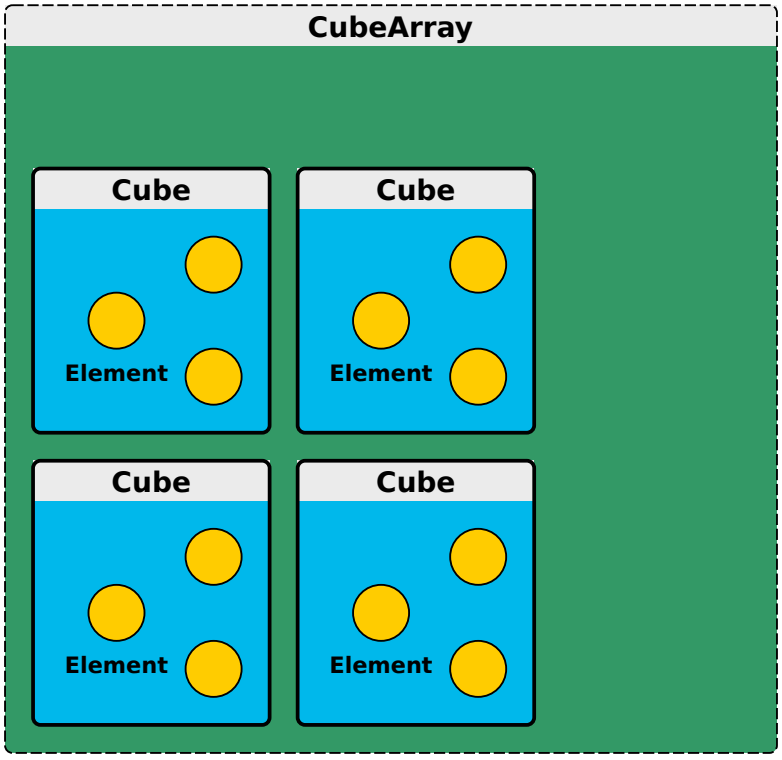
Ο αλγόριθμος χωρίζει το χώρο σε κύβους (βλ. διάγραμμα 1). Αν δεχθούμε ότι δεν υπάρχουν κενοί κύβοι, οι βασικοί υποψήφιοι S κάθε ερωτήματος Q είναι το σύνολο C_{Q1} των σημείων που περιλαμβάνει ο κύβος που περιέχει το Q και όλοι οι γείτονές του. Αν δεχθούμε ότι ενδεχομένως υπάρχουν κενοί κύβοι, τότε με 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,



Σχήμα 2: Ροή προγράμματος



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

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.

Η πορεία του προγράμματος παρουσιάζεται στο διάγραμμα 2.

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:

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

Αλγόριθμος 1 query ()

1. Εύρεση κύβου qloc μέσα στα όρια του οποίου βρίσκεται το Q
2. Χώρος αναζήτησης: searchSpace = qloc
3. search (kNN)
4. Όσο δε βρέθηκαν k σημεία ή η απόσταση του Q από το σύνορο του searchSpace είναι μικρότερη από την απόσταση του πιο απομακρυσμένου kNN
   α′) expand (searchSpace) (συμπερίληψη όλων των κύβων που συνορεύουν με τον τωρινό χώρο αναζήτησης)
   β′) search (kNN)

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, τερματίζοντας την εκτέλεση.

Πιθανότητα εύρεσης γείτονα

Για τη μελέτη της λεπτότητας του καταμερισμού του χώρου που επιτρέπει αυτός ο αλγόριθμος γίνεται μια πιθανοτική ανάλυση.

Έστω καρτεσιανός χώρος Δ στον οποίο τοποθετούνται τυχαία N σημεία και ένας υποχώρος του S. Έστω επίσης X το πλήθος των σημείων που περιέχονται στον S. Τότε η X ακολουθεί διωνυμική κατανομή με παραμέτρους N, $p_\in$ όπου $p_\in$ η πιθανότητα για καθένα από τα σημεία να ανήκει στο S.

Αν στο Δ τα N σημεία τοποθετούνται με ομοιόμορφα τυχαίο τρόπο, τότε $p_\in = \frac{|S|}{|\Delta|}$ (και $p_\notin = 1 - p_\in$).

Ας θεωρήσουμε το Δ ως το χώρο $[0,1)^3$ του προβλήματος. Αν $k < Np_\in$, μπορούμε να φράξουμε τη διωνυμική κατανομή με τη βοήθεια της ανισότητας Chernoff[2]. Η πιθανότητα επαρκών γειτόνων στο S γίνεται τότε:

$$p_s = p\left(X \geq k\right) = 1 - p\left(X < k\right) = 1 - p\left(X \leq K\right) = 1 - F\left(K, N, p_\in\right)$$

Όπου από την ανισότητα Chernoff $F\left(n,p,k\right) \leq exp\left(\frac{-(np-k)^2}{2np}\right)$ έχουμε ένα κάτω φράγμα για την p_s. Αν απαιτήσουμε λοιπόν η πιθανότητα $p_s > 1 - a^{-1}$, όπου a κάποιος μεγάλος αριθμός, μπορούμε να απαιτήσουμε το ίδιο και από το κάτω φράγμα της και οδηγούμαστε στην ανίσωση:

$$p_s \geq 1 - exp\left(\frac{-(Np_\in - K)^2}{2Np_\in}\right) > 1 - \alpha^{-1} \Leftrightarrow exp\left(\frac{-(Np_\in - K)^2}{2Np_\in}\right) < \alpha^{-1} \Leftrightarrow \frac{(Np_\in - K)^2}{2Np_\in} > lna$$

[2]Βλέπε https://en.wikipedia.org/wiki/Binomial_distribution#Tail_Bounds

$$N^2 p_{\in}^2 - 2N(k-1+lna)p_{\in} + (k-1)^2 > 0$$

που είναι ισοδύναμη με την απαίτηση

$$p\left(X \geq k\right) > 1 - a^{-1}$$

Για παράδειγμα, στην περίπτωση που $k = 1$, $N = 2^{25}$, $a = 2^{45}$, δηλαδή η πιθανότητα στο σύνολο των N ερωτημάτων ένα να μην ανήκει στο χώρο αναζήτησης να είναι μικρότερη από περίπου 10^6, σημαίνει ότι ο χώρος αναζήτησης θα πρέπει να έχει μέγεθος

$$|S| = p_{\in} > 1.86 \times 10^{-6}$$

Αν θεωρήσουμε ότι S είναι ο χώρος μετά την 1η επέκταση, άρα αποτελείται από 27 κύβους, τότε $|Cube| > 6.8858 \times 10^{-8} = 2^{-23.8}$. Όμως $|Cube| = (n \times m \times k)^{-1}$, άρα η λεπτότητα του καταμερισμού του χώρου μπορεί με ασφάλεια να φθάσει το 2^{23}. Βέβαια, με λιγότερα από 8 σημεία ανά κύβο, δευτερεύοντα στοιχεία όπως η διεύρυνση του χώρου θα αρχίσουν να κυριεύουν το χρόνο εκτέλεσης.

Η ανάλυση αυτή γενικεύεται και για οποιαδήποτε άλλη γνωστή κατανομή σημείων, μόνο που τότε το $p_{\in}$ θα εξαρτάται από το S κι όχι μόνο από το μέτρο του.

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 test-cases 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

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 $< \times 2$.

Έχουν ληφθεί 2 σετ δεδομένων, από τον τοπικό υπολογιστή και από το cluster. Ο τοπικός υπολογιστής παρέχει 4 φυσικά threads. Το cluster είχε συνωστισμό στη διάρκεια των μετρήσεων, οπότε μέσα σε 2.5 μέρες έγιναν εκτελέσεις με 2 το πολύ nodes – εργασίες που ζητούν περισσότερα περιμένουν ακόμη στην ουρά! Τα δεδομένα αποτελούνται από 3 μετρήσεις χρόνου ανά εκτέλεση:

1. Αίτηση επικοινωνίας για ανταλλαγή σημείων μέχρι παραλαβή σημείων
2. Αίτηση επικοινωνίας για ανταλλαγή σημείων μέχρι παραλαβή και ερωτημάτων (υπερσύνολο του προηγούμενου)

3. Συνολικός χρόνος απάντησης σε όλα τα ερωτήματα

Η μέτρηση 1 περιλαμβάνει μονάχα επικοινωνίες MPI και για αρκετά μεγάλο πρόβλημα όλος ο χρόνος ξοδεύεται στην αναμονή της επικοινωνίας των σημείων C, επομένως είναι ενδεικτική της ποιότητας των αργότερων επικοινωνιών στο σύστημα. Όλες οι εκτελέσεις στο grid περιλαμβάνουν 2 nodes. Η μέτρηση 2 δεν έχει ξεκάθαρο περιεχόμενο (λόγω επικάλυψης επεξεργασίας με τις nonblocking επικοινωνίες) και συλλέχθηκε συμπληρωματικά. Η μέτρηση 3 αποτελεί το μεγαλύτερο τμήμα του συνολικού χρόνου εκτέλεσης.

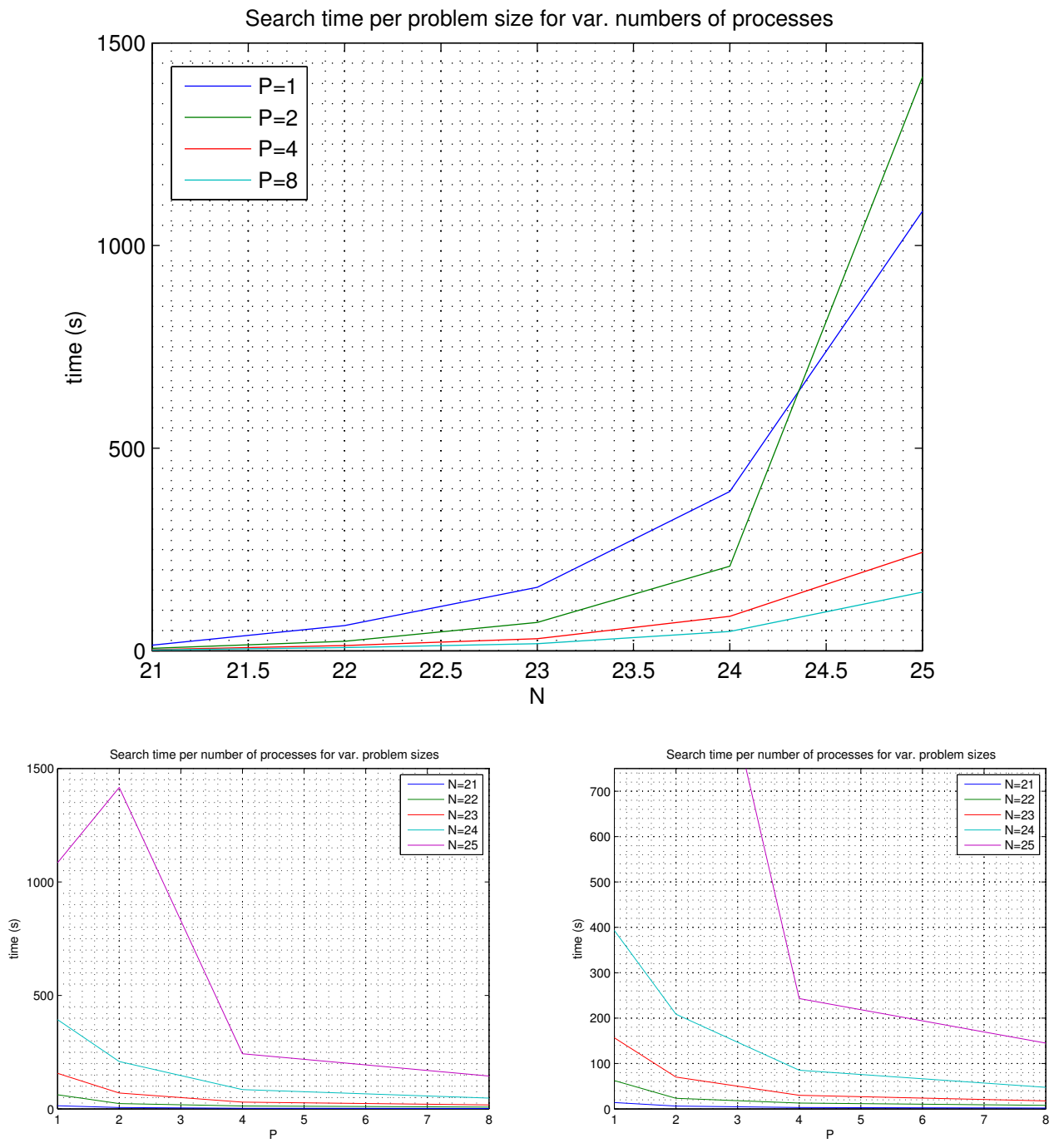
Τα δεδομένα που συλλέχθηκαν από τον τοπικό υπολογιστή καλύπτουν το εύρος των παραμέτρων N και $n \times m \times k$ σε σειριακή και 4-process εκτέλεση. Εξαιτίας του συνωστισμού στο cluster συλλέχθηκαν από εκεί επιλεγμένα δεδομένα, με σταθερή την παράμετρο $n \times m \times k$ και σαρώνοντας το μέγεθος του προβλήματος και το πλήθος των διεργασιών. Οι μετρήσεις έφθασαν μέχρι 2^3 μόνο διεργασίες, αλλά στον τοπικό υπολογιστή επιβεβαιώθηκε ότι το πρόγραμμα λειτουργεί χωρίς σφάλματα μέχρι και για 2^7 διεργασίες – μάλιστα, με ελαφρά βελτιωμένη απόδοση σε σχέση με τις 4 διεργασίες.

Παρατηρήσεις μετρήσεων

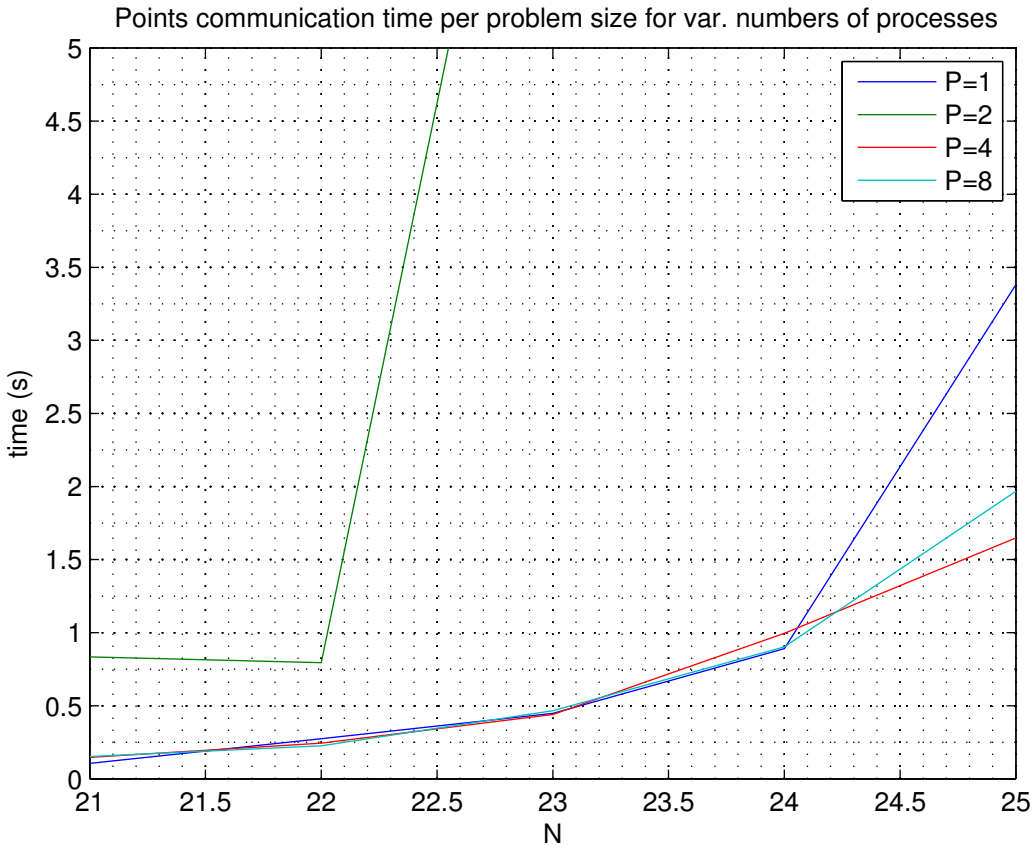
- Το πρόβλημα έχει τετραγωνική πολυπλοκότητα, όπως δείχνουν τα γραφήματα 3 και 6. Η μέτρηση για $P = 2$ στο cluster είναι προβληματική, όπως φαίνεται σε όλα τα γραφήματα, και αυτό μπορεί να οφείλεται σε αλληλεπίδραση με κάποια άλλη ταυτόχρονη εργασία του cluster.

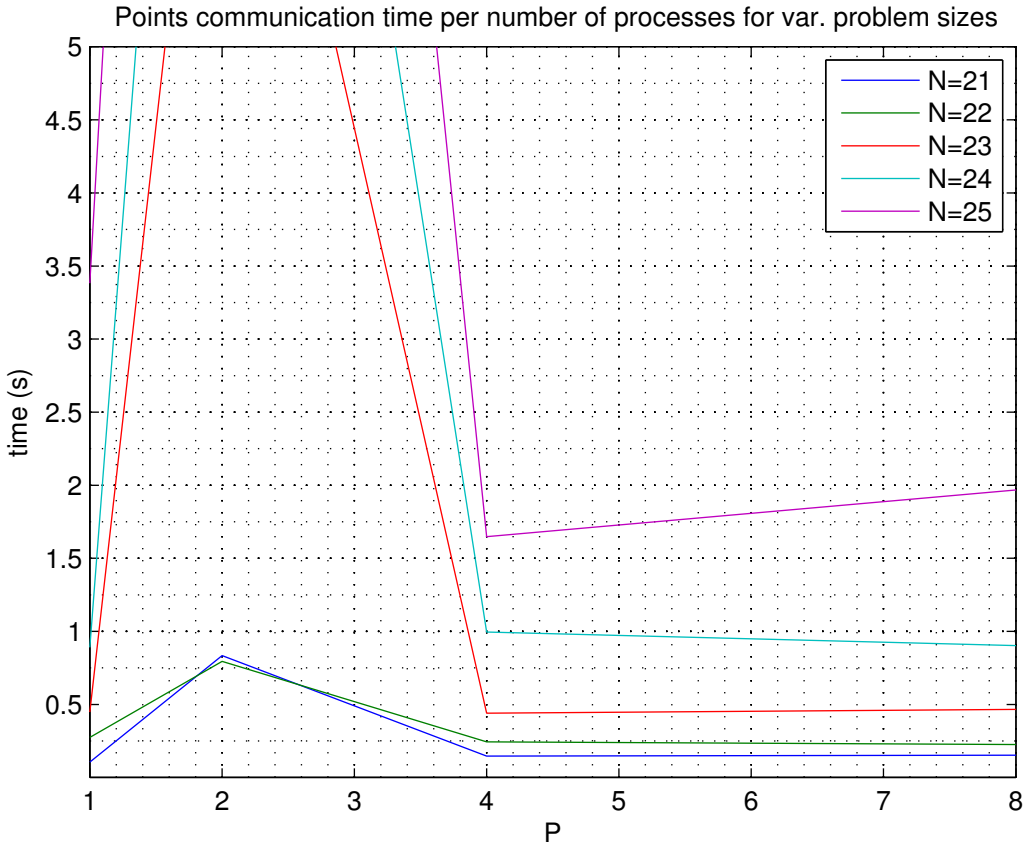
- **Η παραλληλοποίηση είναι σχεδόν τέλεια**, όπως εξάλλου αναμένεται από την έλλειψη αλληλεπίδρασης των διεργασιών μετά από τις αρχικές επικοινωνίες, και αυτό φαίνεται στο 3ο υπογράφημα των γραφημάτων 3. Παρόλα αυτά, οι μετρήσεις στο laptop δείχνουν πολύ λιγότερη επιτάχυνση από την παραλληλοποίηση, της τάξης του $x2$ για 4 διεργασίες ως προς το σειριακό, που μάλλον οφείλεται στην ταυτόχρονη χρήση του υπολογιστή από άλλες διεργασίες και τα αναπόφευκτα context switch.

- Ο καταμερισμός του χώρου σε κύβους έχει πολύ μεγάλη επίδραση στο χρόνο εκτέλεσης, γεγονός που δικαιολογεί την προσέγγιση ιδιαίτερα λεπτού καταμερισμού και διαδοχικών διευρύνσεων του χώρου αναζήτησης που ακολουθείται εδώ. Ακόμη και για καταμερισμό του επιπέδου $\frac{N}{n \times m \times k} = 2^3$, δηλαδή με αναμενόμενο αριθμό σημείων ανά κύβο μόλις 8, η πιθανότητα να μην υπάρχει γείτονας μετά την πρώτη επέκταση για $N = 2^{25}$ είναι μικρότερη από 2^{45}!

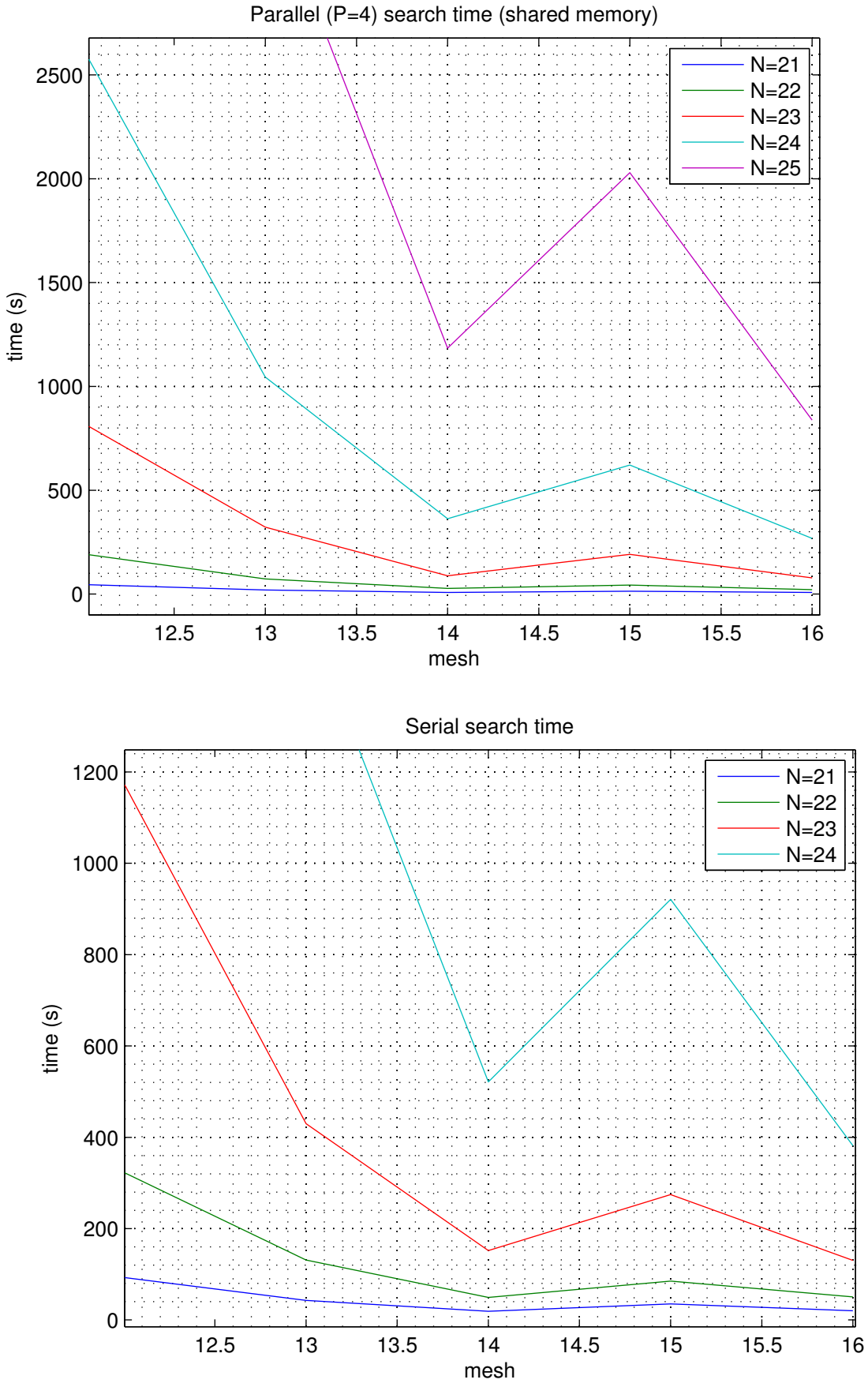


Σχήμα 3: Cluster – Χρόνος kNN σε συνάρτηση με μέγεθος προβλήματος και πλήθος διεργασιών

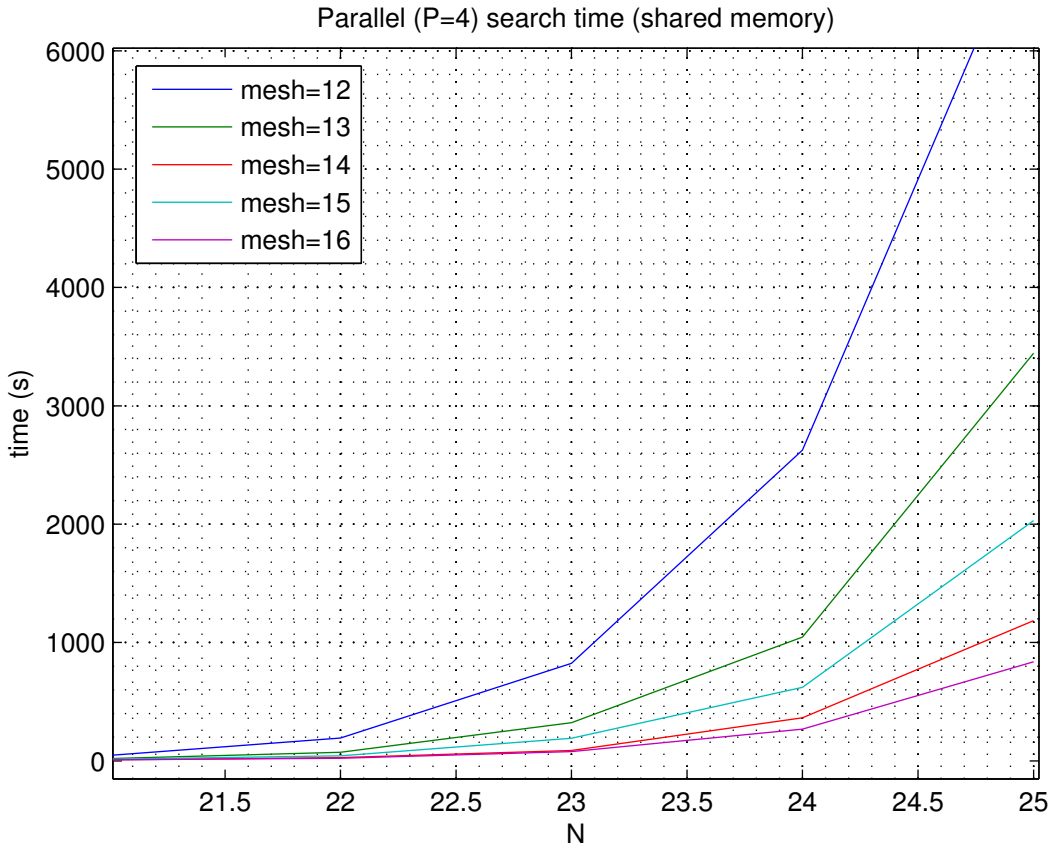


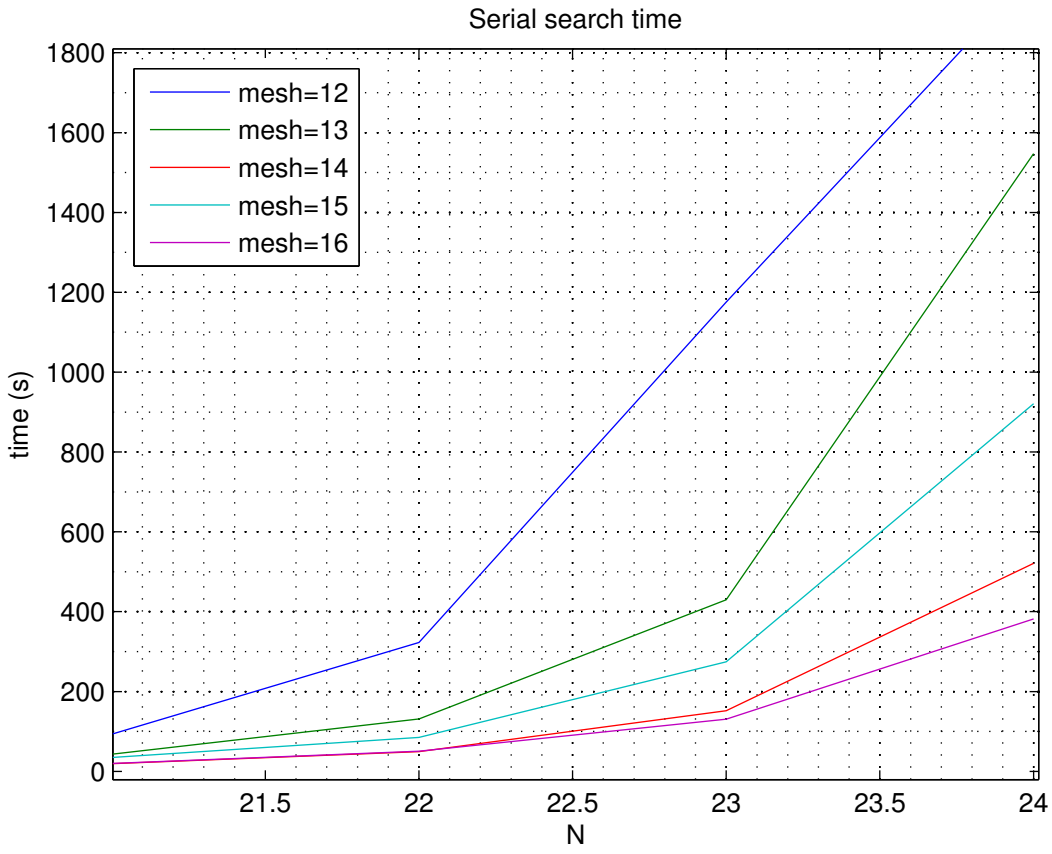


Σχήμα 4: Cluster – Χρόνος επικοινωνιών σε συνάρτηση με μέγεθος προβλήματος και πλήθος διεργασιών

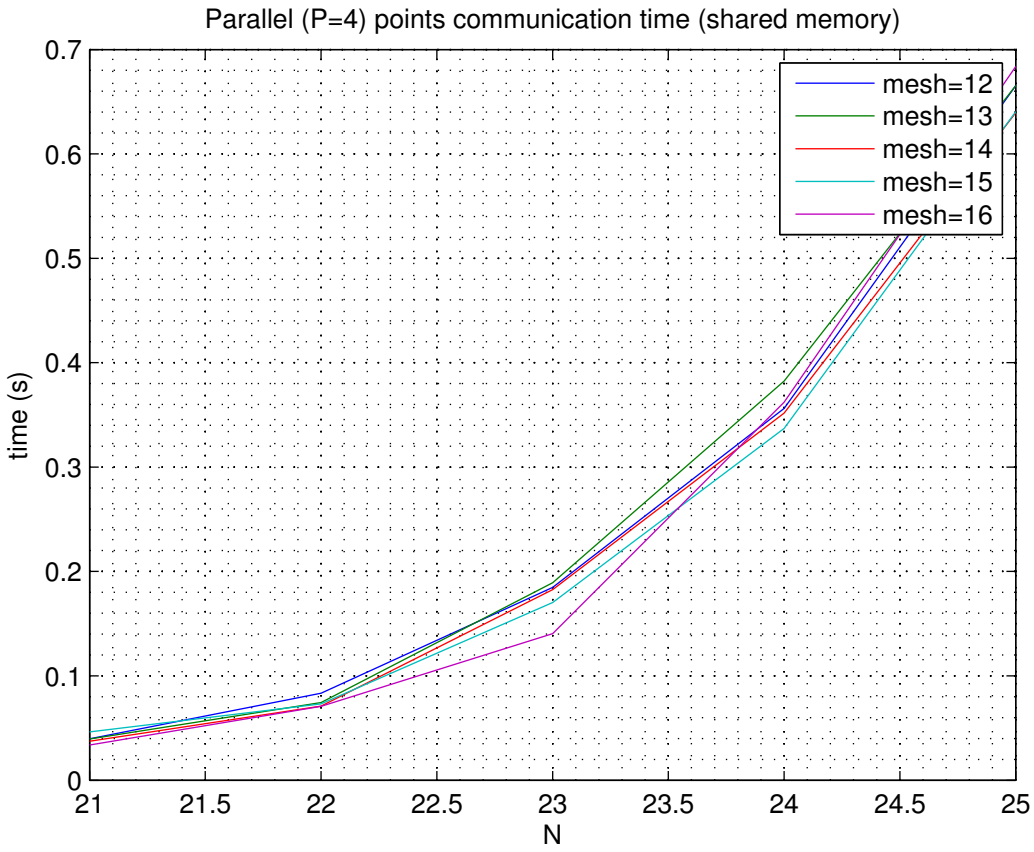


Σχήμα 5: Laptop – Χρόνος αναζήτησης ως προς πλήθος κύβων





Σχήμα 6: Laptop – Χρόνος αναζήτησης ως προς μέγεθος προβλήματος – Προσοχή στα όρια του x άξονα!


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