ΜΥΥ023-ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018

2ο ΣΕΤ ΑΣΚΗΣΕΩΝ

Μπουρλή Στυλιανή
ΑΜ: 2774
email: stellabourli96@gmail.com

Πληροφορίες για το σύστημα που χρησιμοποιήθηκε:

Όνομα υπολογιστή	opti7020ws10
Επεξεργαστής	Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
Πλήθος πυρήνων	4
Μεταφραστής	gcc (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4

1η Άσκηση

"Αρχεία κώδικα σε γλώσσα C: mat1.c, mat2.c, mat3.c"

Στην άσκηση αυτή ζητήθηκε η παραλληλοποίηση του σειριακού προγράμματος πολλαπλασιασμού τετραγωνικών πινάκων με τη χρήση του OpenMP. Συγκεκριμένα, ο κάθε βρόγχος του σειριακού προγράμματος έπρεπε να παραλληλοποιηθεί και να χρονομετρηθεί ξεχωριστά 2 φορές, τη μία με στατική διαμοίραση εργασιών στα νήματα και την άλλη με δυναμική.

Υλοποίηση

Για την παραλληλοποίηση του κάθε βρόγχου δημιούργησα ένα ξεχωριστό αρχείο κώδικα.

1ος βρόγχος (αρχείο mat1.c)
Για τον πρώτο βρόγχο δημιούργησα μία παράλληλη περιοχή που τον περικλείει και όρισα τον τύπο της κάθε μεταβλητής ,καθώς και το πλήθος των νημάτων. Πιο συγκεκριμένα, όρισα τις μεταβλητές i, j, k, sum ως private, για να είναι ιδιωτικές για κάθε νήμα, τους πίνακες A, B, C ως shared, για να είναι κοινόχρηστοι και το πλήθος των νημάτων σε 4. Επιπρόσθετα, για την παραλληλοποίηση του βρόγχου χρησιμοποίησα το schedule, για αυτόματη διαμοίραση εργασιών στα νήματα, δίνοντάς του το όρισμα runtime. Με αυτό τον τρόπο καθόριζα τον τρόπο διαμοίρασης (στατικό / δυναμικό) σε χρόνο εκτέλεσης, χρησιμοποιώντας μία φορά *export OMP_SCHEDULE "static"* και μία φορά *export OMP_SCHEDULE "dynamic"*.

2ος βρόγχος (αρχείο mat2.c)
Για το δεύτερο βρόγχο δημιούργησα μία παράλληλη περιοχή που τον περικλείει και όρισα τον τύπο της κάθε μεταβλητής, καθώς και το πλήθος των νημάτων. Πιο συγκεκριμένα, όρισα τις μεταβλητές j, k, sum ως private, για να είναι ιδιωτικές για κάθε νήμα, ενώ τη μεταβλητή i και τους πίνακες A, B, C ως shared. Το πλήθος των νημάτων και σε αυτή την περίπτωση το όρισα ίσο με 4. Επίσης, για την παραλληλοποίηση του βρόγχου χρησιμοποίησα το schedule, για αυτόματη διαμοίραση εργασιών στα νήματα, δίνοντάς του το όρισμα runtime. Με αυτό τον τρόπο καθόριζα τον τρόπο διαμοίρασης (στατικό / δυναμικό) σε χρόνο εκτέλεσης, χρησιμοποιώντας μία φορά *export OMP_SCHEDULE "static"* και μία φορά *export OMP_SCHEDULE "dynamic"*. Τέλος, όρισα την περιοχή που προστίθεται το αποτέλεσμα του πολλαπλασιασμού στον πίνακα C ως κρίσιμη, γιατί σε αυτήν εμπλέκεται η κοινόχρηστη μεταβλητή i.

3ος βρόγχος (αρχείο mat3.c)
Για τον τρίτο βρόγχο δημιούργησα μία παράλληλη περιοχή που τον περικλείει και όρισα τον τύπο της κάθε μεταβλητής, καθώς και το πλήθος των νημάτων. Πιο συγκεκριμένα, όρισα τις μεταβλητές k και sum ως private, για να είναι ιδιωτικές για κάθε νήμα, ενώ τις μεταβλητές i, j και τους πίνακες A, B, C ως shared. Το πλήθος των νημάτων και σε αυτή την περίπτωση το όρισα ίσο με 4. Επίσης, για την παραλληλοποίηση του βρόγχου χρησιμοποίησα το schedule, για αυτόματη διαμοίραση εργασιών στα νήματα, δίνοντάς του το όρισμα runtime. Με αυτό τον τρόπο καθόριζα τον τρόπο διαμοίρασης (στατικό / δυναμικό) σε χρόνο εκτέλεσης, χρησιμοποιώντας μία φορά *export OMP_SCHEDULE "static"* και μία φορά *export OMP_SCHEDULE "dynamic"*. Τέλος, όρισα την περιοχή που προστίθεται το αποτέλεσμα του πολλαπλασιασμού στον πίνακα C ως κρίσιμη, γιατί σε αυτήν εμπλέκονται οι κοινόχρηστες μεταβλητές i και j.

Χρονομέτρηση

Για τη χρονομέτρηση χρησιμοποίησα σε όλες τις περιπτώσεις κλήσεις χρονομέτρησης που παρέχει το OpenMP *(omp_get_wtime())*.

Κάθε πρόγραμμα (παραλληλοποίηση ξεχωριστού βρόγχου) το έτρεξα 4 φορές για στατική διαμοίραση εργασιών και 4 φορές για δυναμική.

Όλες οι μετρήσεις φαίνονται στους παρακάτω πίνακες:

Σειριακό πρόγραμμα

1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος Όρος**
3.866293	4.778142	3.921331	3.921331	4.12177425

1ος βρόγχος – στατική διαμοίραση (static)

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.782405	0.786405	0.779878	0.787981	0.78416725

1ος βρόγχος – δυναμική διαμοίραση (dynamic)

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.935472	0.783392	0.791294	0.790596	0.8251885

2ος βρόγχος – στατική διαμοίραση (static)

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.819154	1.044256	1.043377	1.047391	0.9885445

2ος βρόγχος – δυναμική διαμοίραση (dynamic)

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.927490	0.813339	1.125396	1.068259	0.983621

3ος βρόγχος – στατική διαμοίραση (static)

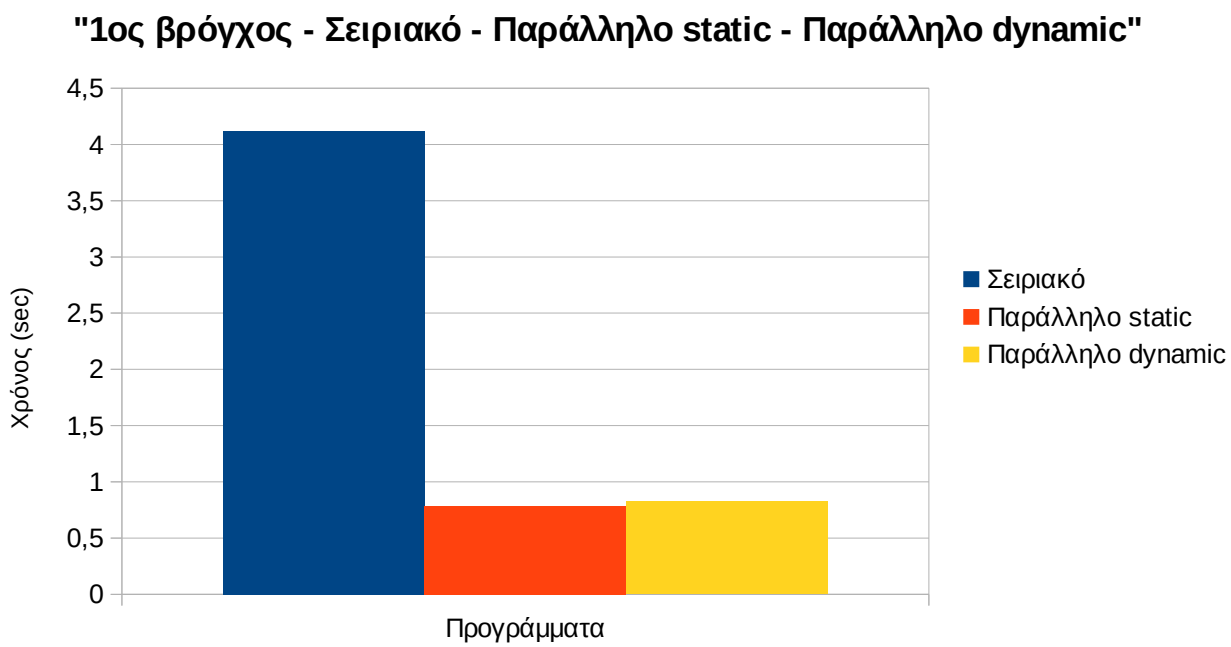
1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
30.453536	30.762966	30.290330	30.810913	30.57943625

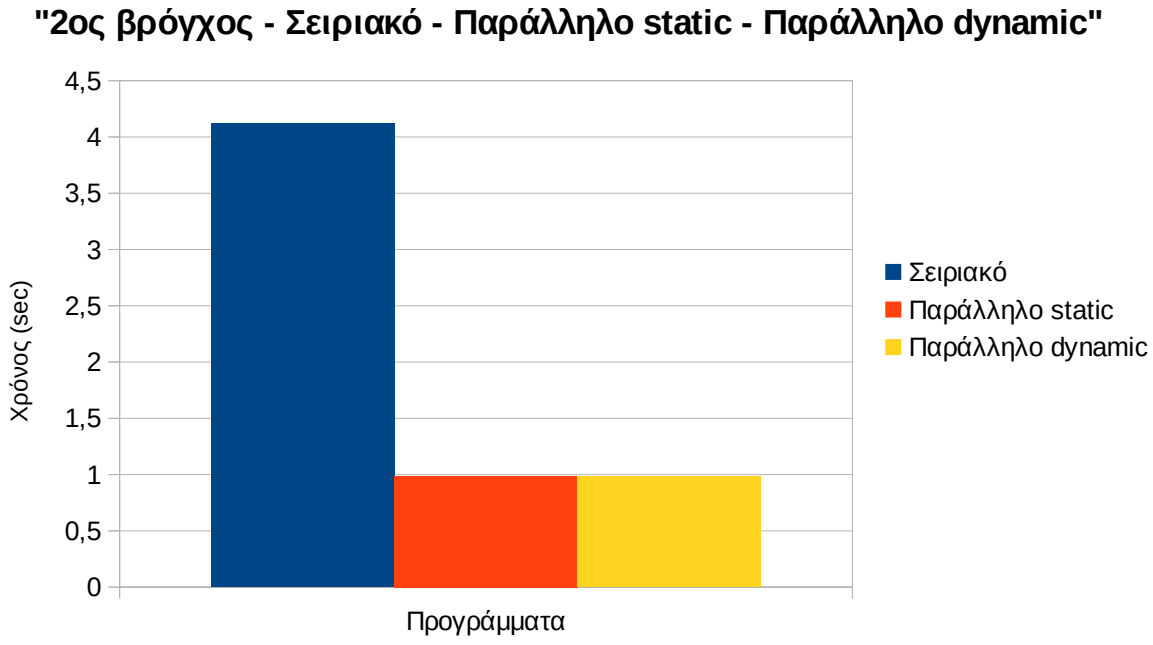
3ος βρόγχος – δυναμική διαμοίραση (dynamic)

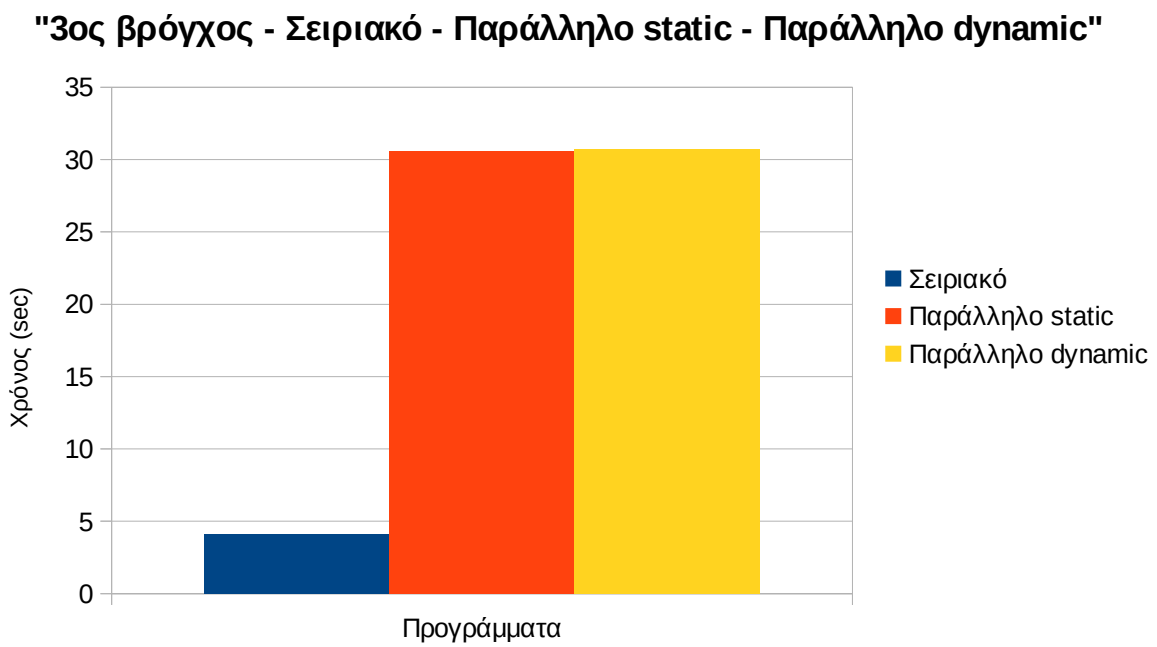
1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
30.816820	30.685415	30.659586	30.705626	30.71686175

Γραφικές Παραστάσεις

Με βάση τις παραπάνω μετρήσεις δημιούργησα τις εξής γραφικές παραστάσεις:







Παρατηρήσεις

1ος βρόγχος: Η παραλληλοποίηση του 1ου βρόγχου μείωσε δραματικά το χρόνο εκτέλεσης. Ελάχιστα περισσότερη μείωση του χρόνου πέτυχε η παραλληλοποίηση με στατική διαμοίραση του χρόνου.

2ος βρόγχος: Η παραλληλοποίηση του 2ου βρόγχου μείωσε και αυτή αρκετά το χρόνο εκτέλεσης, όχι όμως στο βαθμό της παραλλήλοποίησης του 1ου βρόγχου. Λίγο καλύτερη σε αυτή την περίπτωση ήταν η δυναμική διαμοίραση εργασιών.

3ος βρόγχος: Η παραλληλοποίηση του 3ου βρόγχου δεν βοήθησε στη μείωση του χρόνου εκτέλεσης. Αντίθετα ο χρόνος εκτέλεσης αυξήθηκε πολύ σε σχέση με το χρόνο του σειριακού προγράμματος.

2η Άσκηση

“Αρχείο κώδικα σε γλώσσα C: primes.c”

Στην άσκηση αυτή μας δόθηκε ένα πρόγραμμα στο οποίο υπήρχε μία συνάρτηση, η serial_primes(), η οποία υπολόγιζε, δεδομένου ενός αριθμού N, το πλήθος των πρώτων αριθμών καθώς και το μεγαλύτερο πρώτο αριθμό μέχρι και το N. Στο πρόγραμμα υπήρχε μία ακόμη συνάρτηση, η openmp_primes(), την οποία μας ζητήθηκε να συμπληρώσουμε με την παράλληλη εκδοχή του αλγορίθμου της serial_primes() και χρήση του OpenMP.

Λογική Υλοποίηση

Για να καταλήξω στην κατάλληλη παράλληλη υλοποίηση, αλλά και στο κατάλληλο πλήθος νημάτων, δοκίμασα διαφορετικούς τρόπους υλοποίησης και διαφορετικό πλήθος νημάτων για κάθε τρόπο. Συγκεκριμένα, δοκίμασα από ένα έως και τέσσερα νήματα. Έπειτα, χρονομέτρησα κάθε περίπτωση και έκανα τις κατάλληλες συγκρίσεις καταλήγοντας στον παρακάτω τρόπο υλοποίησης,

Υλοποίηση

Αρχικά, όρισα μία παράλληλη περιοχή, που περικλείει το βρόγχο του αλγορίθμου. Όρισα, στη συνέχεια, τις μεταβλητές i, num, divisor, quotient και remainder ως private, για να είναι ιδιωτικές για κάθε νήμα, τις μεταβλητές count και lastprime ως shared, για να είναι κοινόχρηστες και τη μεταβλητή n ως firstprivate, για να είναι ιδιωτική αλλά και αρχικοποιημένη στην αρχική προκαθορισμένη τιμή. Επιπλέον, το πλήθος των νημάτων το έθεσα να ισούται με 4 και η διαμοίραση των εργασιών να γίνεται με guided τρόπο, χρησιμοποιώντας chunk = 1000000 και nowait, ώστε να μην υπονοείται barrier στο τέλος. Τέλος, όρισα ως κρίσιμη την περιοχή count++; lastprime = num; γιατί περιλαμβάνει την τροποποίηση των κοινόχρηστων μεταβλητών.

Παρακάτω φαίνονται οι τρόποι υλοποίησης που δοκίμασα, οι χρόνοι εκτέλεσης του καθενός, κάποιες γραφικές παραστάσεις με βάση τις χρονομετρήσεις και τέλος οι παρατηρήσεις, όπου υπάρχει η σύγκριση τους και η επιλογή του καλύτερου.

Χρονομέτρηση

Για τη χρονομέτρηση χρησιμοποίησα σε όλες τις περιπτώσεις κλήσεις χρονομέτρησης που παρέχει το OpenMP *(omp_get_wtime())*.

Αρχικά, χρονομέτρησα το σειριακό πρόγραμμα, ώστε να το συγκρίνω με τις δικές μου παράλληλες υλοποιήσεις.

Σειριακό

1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
14.143977	14.142494	14.133999	14.938703	14.33979325

Ο πρώτος τρόπος που δοκίμασα είναι να έχω μία παράλληλη περιοχή που να περικλείει το βρόγχο του αλγορίθμου, με τις κατάλληλες δηλώσεις μεταβλητών, χωρίς συγκεκριμένο τρόπο διαμοίρασης των εργασιών.

1ος τρόπος

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.920625	13.912499	13.918085	14.707876	14.11477125
2 νήματα	8.696341	9.195184	8.696615	8.709904	8.824511
3 νήματα	6.464125	6.496231	6.500016	6.437563	6.47448375
4 νήματα	5.937641	5.885246	5.631951	5.658189	5.77825675

Ο δεύτερος τρόπος που δοκίμασα είναι να έχω μία παράλληλη περιοχή που να περικλείει το βρόγχο του αλγορίθμου, με τις κατάλληλες δηλώσεις μεταβλητών, με στατικό τρόπο διαμοίρασης των εργασιών.

2ος τρόπος

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.919667	13.915634	13.916333	13.917343	13.91724425
2 νήματα	8.698676	8.704586	8.704244	8.694405	8.70047775
3 νήματα	6.165536	6.163947	6.163947	6.171850	6.16632
4 νήματα	4.803051	4.802843	4.797344	4.809009	4.80306175

Ο τρίτος τρόπος που δοκίμασα είναι να έχω μία παράλληλη περιοχή που να περικλείει το βρόγχο του αλγορίθμου, με τις κατάλληλες δηλώσεις μεταβλητών, με δυναμικό τρόπο διαμοίρασης των εργασιών.

3ος τρόπος

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.001675	12.999432	13.061511	12.999884	13.0156255
2 νήματα	6.523538	6.526790	6.525878	6.526407	6.52565325
3 νήματα	4.472098	4.478333	4.474686	4.474531	4.474912
4 νήματα	3.451048	3.452952	3.450270	3.451538	3.451452

Ο τέταρτος τρόπος που δοκίμασα είναι να έχω μία παράλληλη περιοχή που να περικλείει το βρόγχο του αλγορίθμου, με τις κατάλληλες δηλώσεις μεταβλητών, με guided τρόπο διαμοίρασης των εργασιών.

4ος τρόπος

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	12.960062	12.961807	12.961807	12.959608	12.960821
2 νήματα	6.489079	6.500272	6.494602	6.493156	6.49427725
3 νήματα	4.447744	4.455983	4.449224	4.451995	4.4512365
4 νήματα	4.447298	4.450377	4.451854	4.450089	4.4499045

Ο πέμπτος τρόπος που δοκίμασα είναι να έχω μία παράλληλη περιοχή που να περικλείει το βρόγχο του αλγορίθμου, με τις κατάλληλες δηλώσεις μεταβλητών, με static, dynamic και guided τρόπο διαμοίρασης των εργασιών και χρήση nowait σε κάθε περίπτωση.

5ος τρόπος

static - nowait

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.087114	13.088686	13.086628	13.086301	13.08718225
2 νήματα	6.565098	6.575923	6.573106	6.568787	6.5707285
3 νήματα	4.506128	4.507525	4.504801	4.506030	4.506121
4 νήματα	3.479110	3.477139	3.474617	3.481430	3.478074

4 νήματα, chunk = 1000000	5.343404	5.346301	5.346962	5.354260	5.34773175
4 νήματα, chunk = 10000	3.677694	3.677541	3.681150	3.681464	3.67946225

dynamic - nowait

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.088803	13.084890	13.082942	13.084216	13.08521275
2 νήματα	6.564754	6.569127	6.572771	6.571400	6.569513
3 νήματα	4.501635	4.505608	4.504690	4.507721	4.5049135
4 νήματα	3.475327	3.493782	3.474342	3.476001	3.479863

4 νήματα, chunk = 1000000	4.995696	4.994773	4.995351	4.996331	4.99553775
4 νήματα, chunk = 10000	3.451329	3.445980	3.446260	3.447180	3.44768725

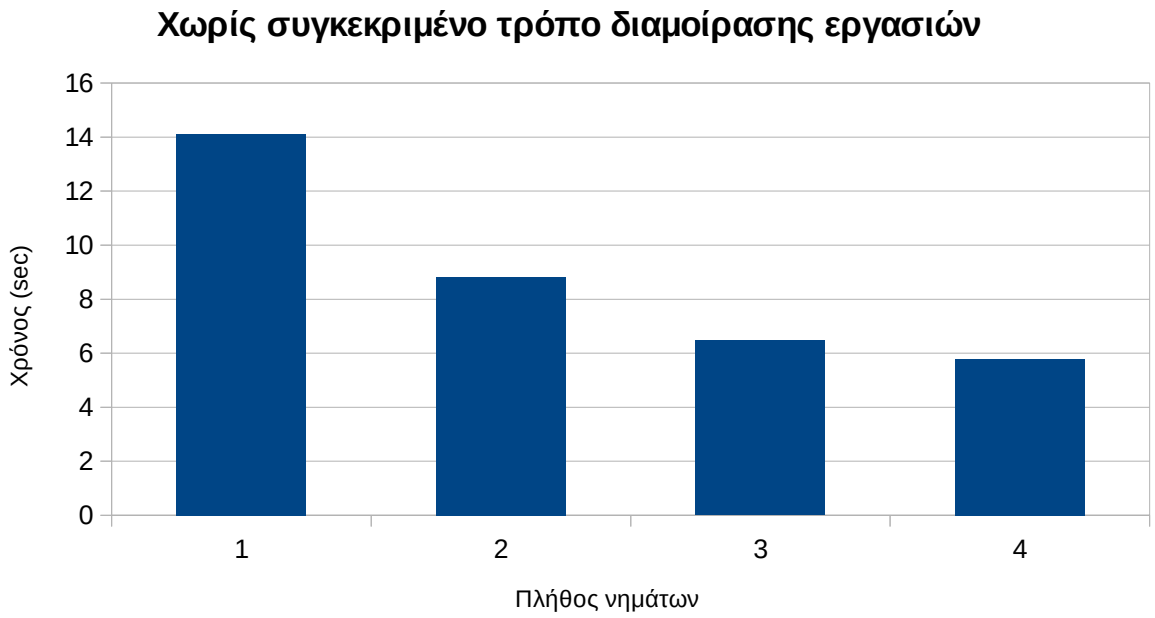
guided - nowait

	1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος όρος**
1 νήμα	13.088508	13.092849	13.087116	13.084234	13.08817675
2 νήματα	6.564041	6.574369	6.572059	6.572259	6.570682
3 νήματα	4.501534	4.505022	4.508392	4.507167	4.50552875
4 νήματα	3.473541	3.474757	3.475916	3.480786	3.47625

4 νήματα, chunk = 1000000	4.649862	4.650644	4.648542	4.650516	4.649891
4 νήματα, chunk = 10000	3.442175	3.440676	3.441361	3.447170	3.4428455

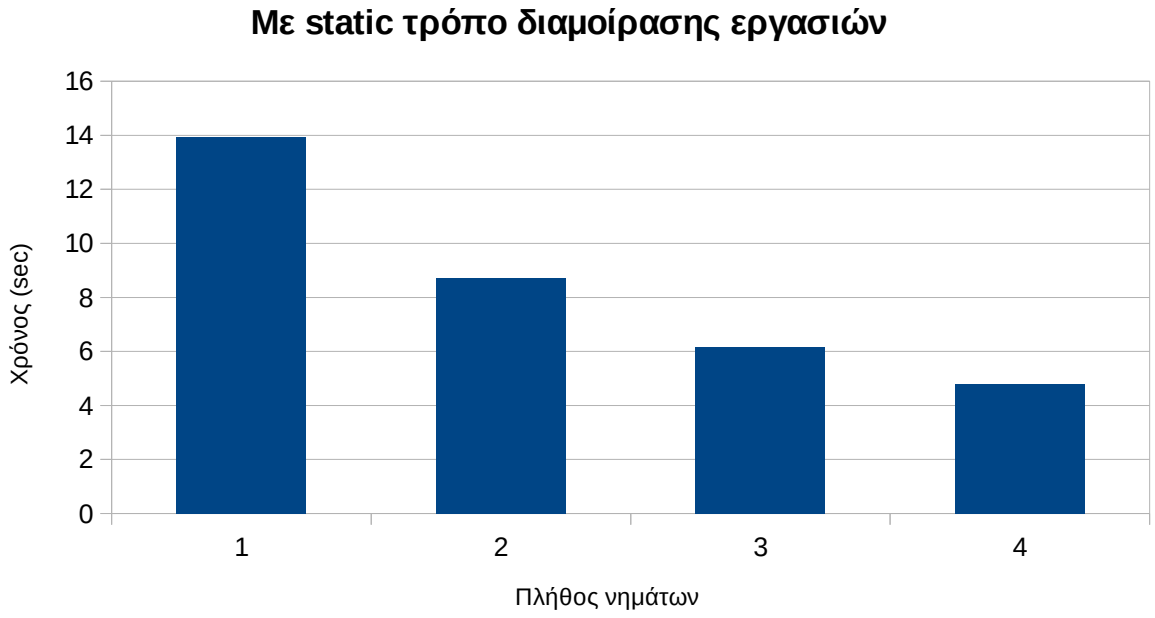
Γραφικές Παραστάσεις – Παρατηρήσεις

1ος τρόπος



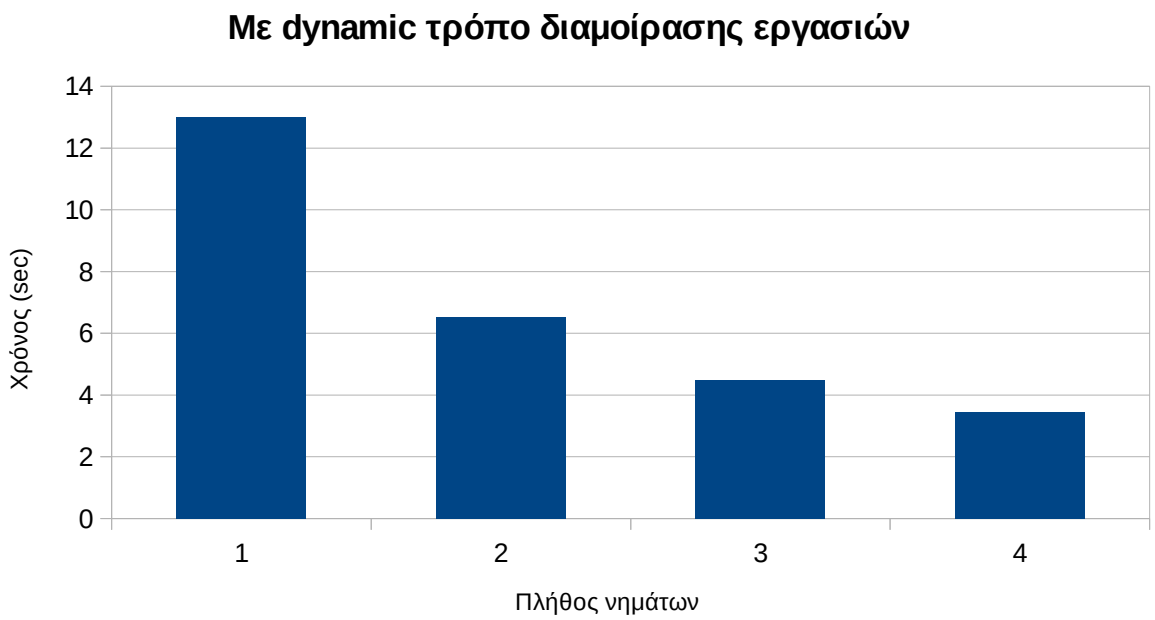
Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα.

2ος τρόπος



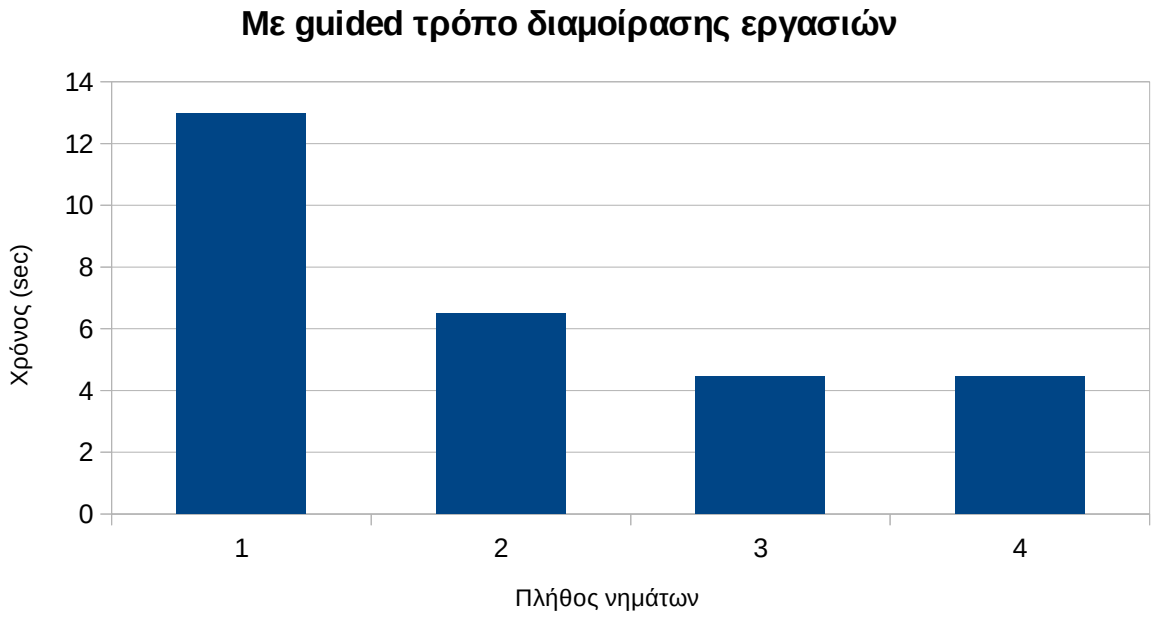
Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα.

3ος τρόπος



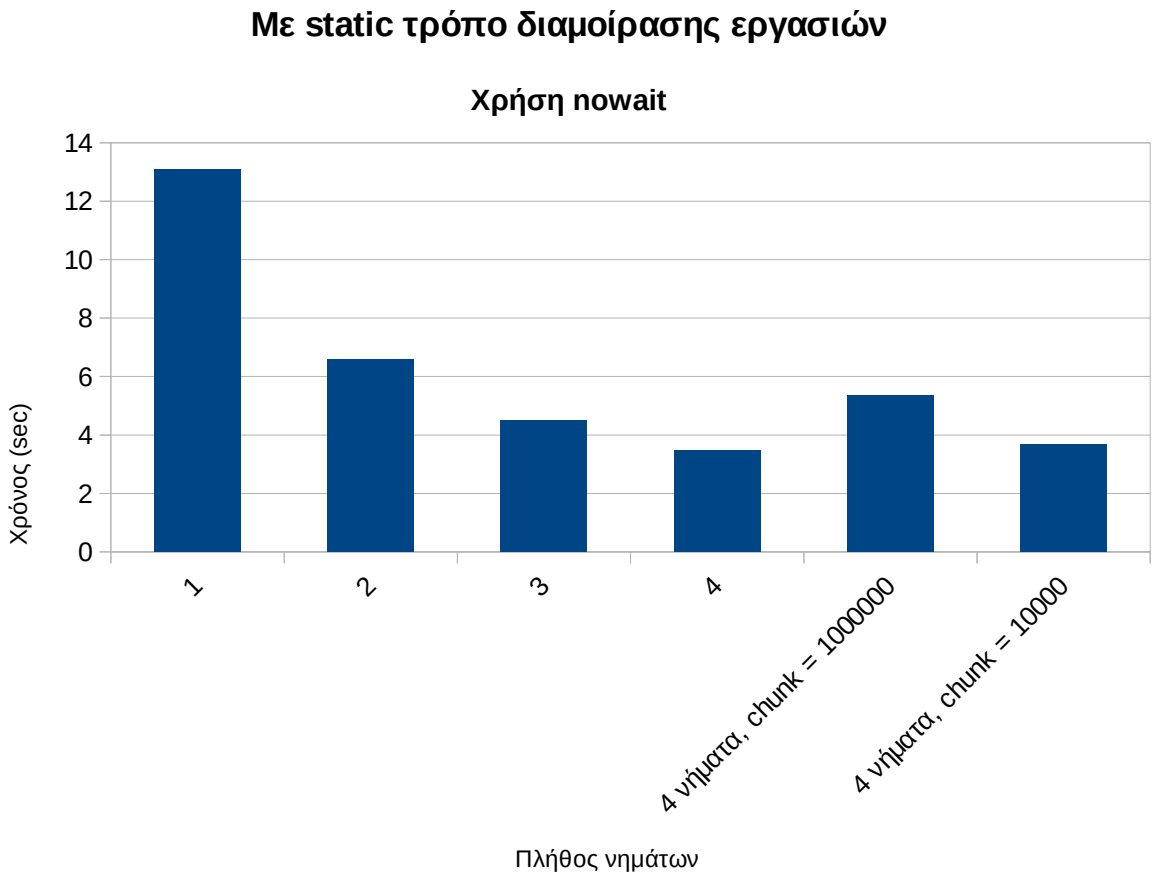
Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα.

4ος τρόπος

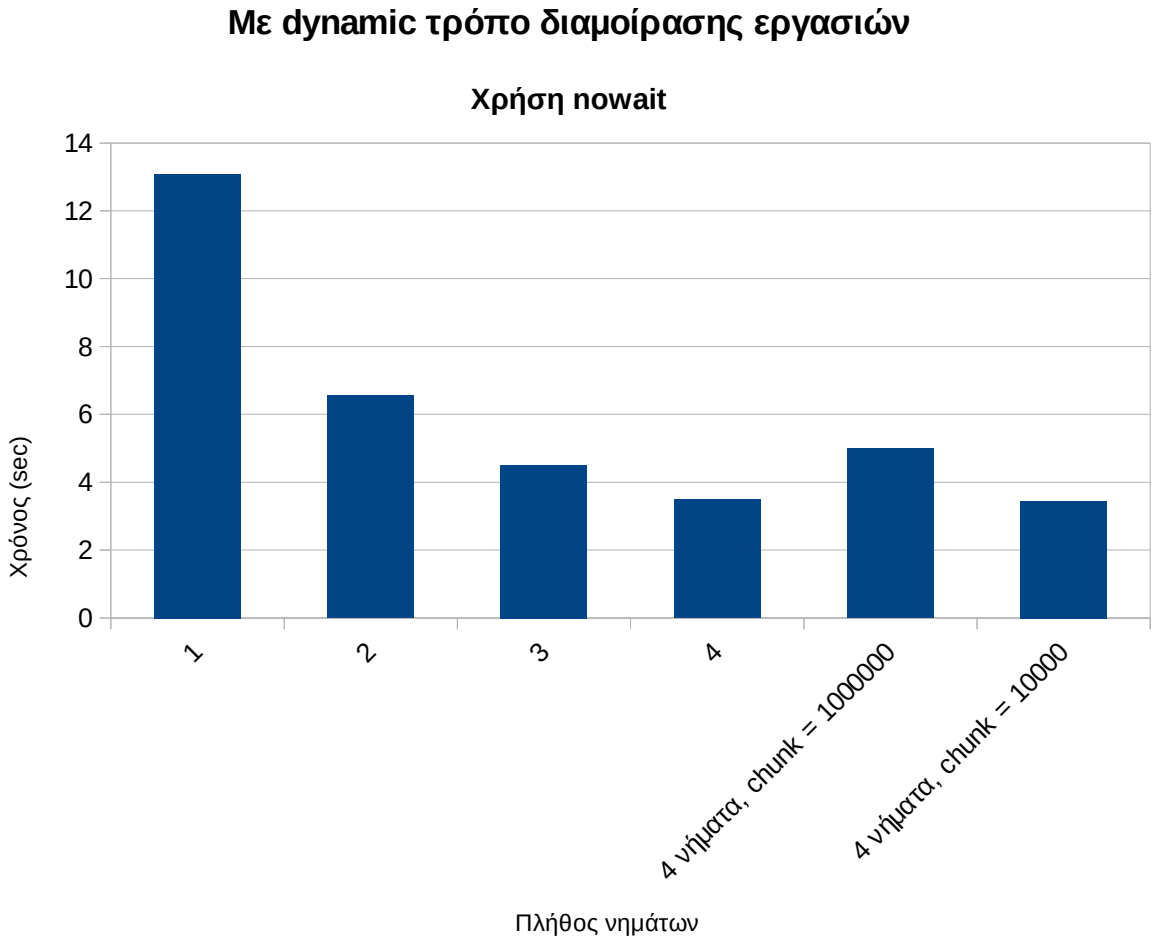


Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα.

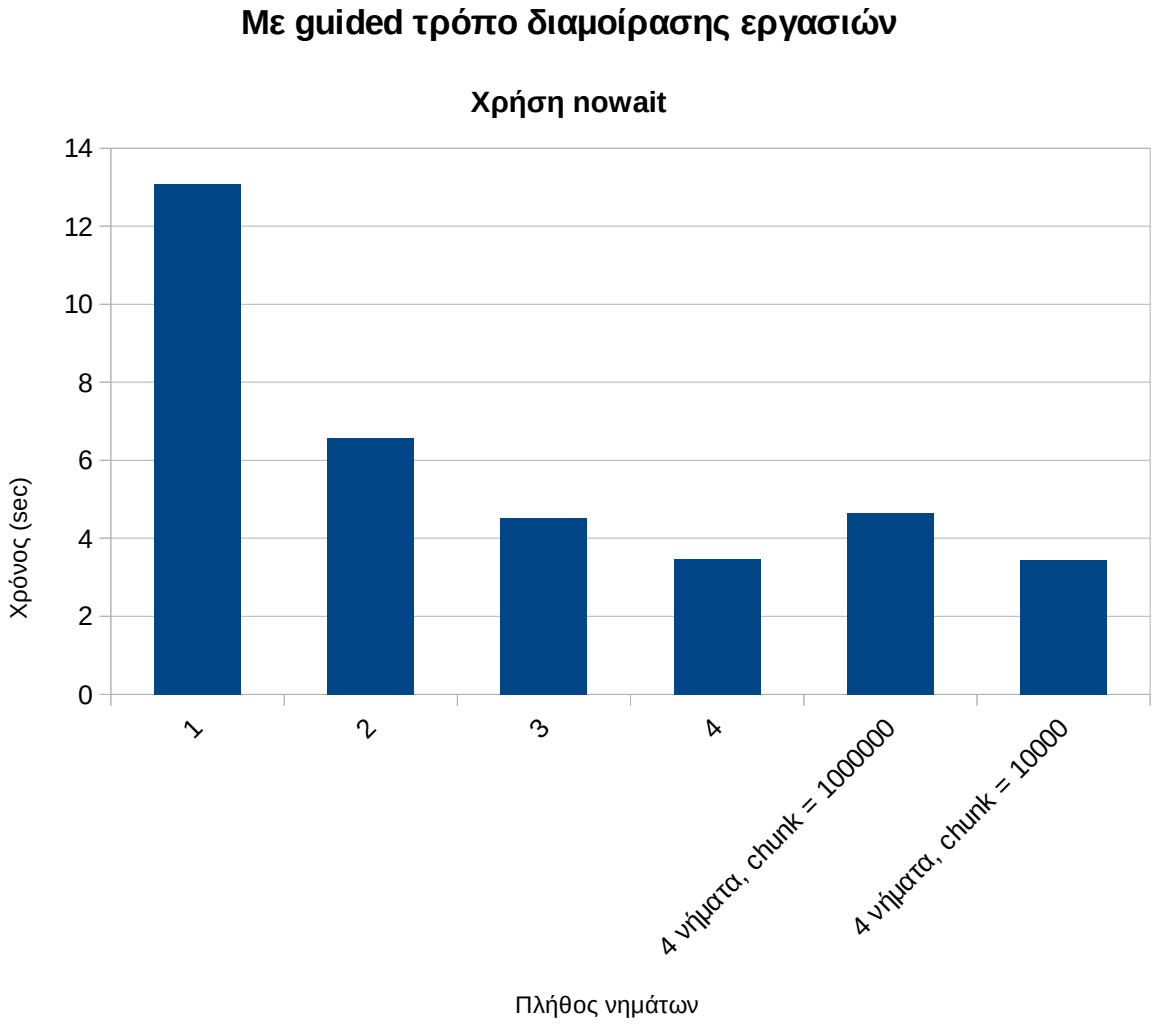
5ος τρόπος



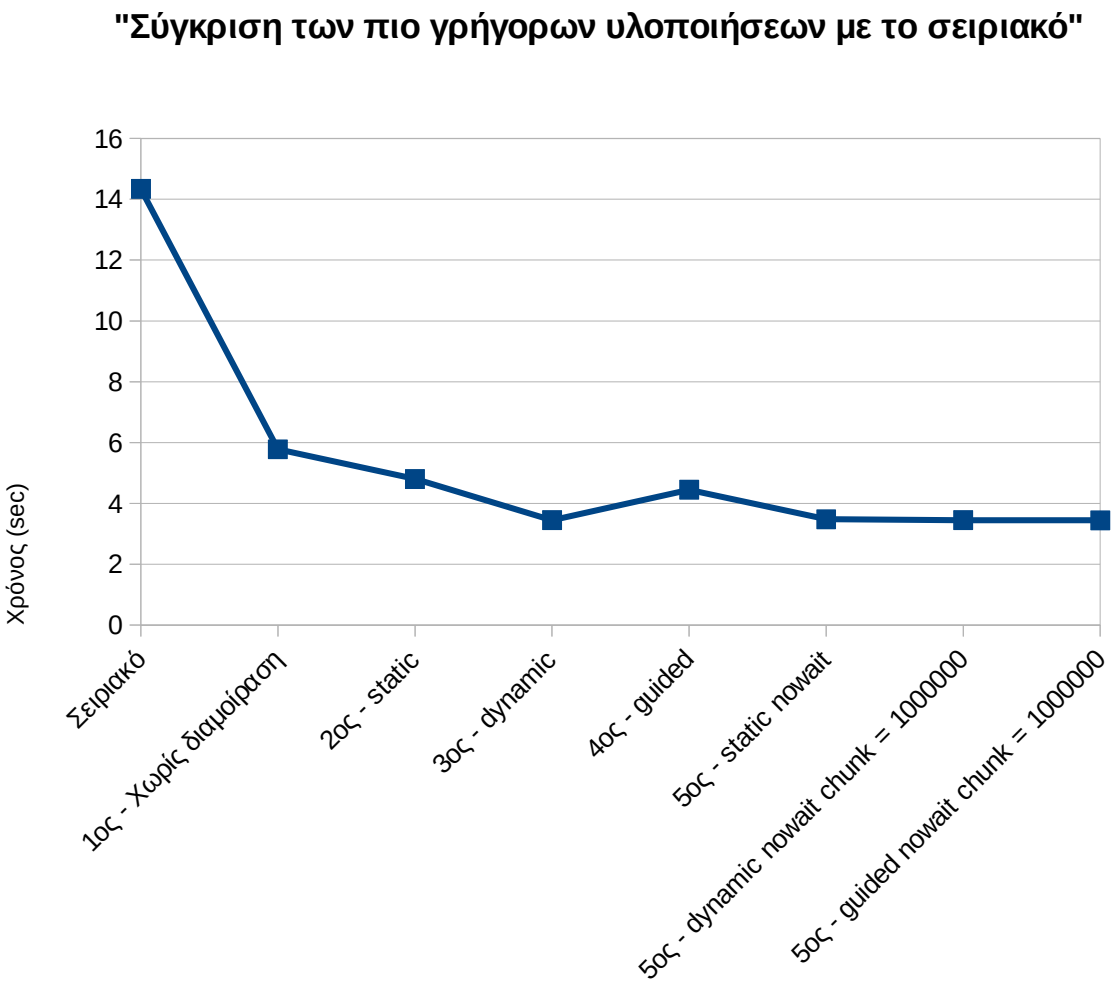
Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα, χωρίς χρήση συγκεκριμένου chunk (αυτόματα chunk = N/4).



Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα και χρήση chunk = 10000 (10000000/1000).



Παρατήρηση: Παρατηρώ ότι πιο γρήγορος είναι ο τρόπος εκτέλεσης με τα 4 νήματα και χρήση chunk = 10000 (10000000/1000).



Παρατήρηση: Παρατηρώ ότι πιο γρήγορη είναι η υλοποίηση με 4 νήματα, guided τρόπο διαμοίρασης και chunk = 1000000.

3η Άσκηση

“Αρχεία κώδικα σε γλώσσα C: multiply_serial.c, multiply_tasks.c”

Σ’ αυτή την άσκηση ζητήθηκε να υλοποιήσουμε και να χρονομετρήσουμε παράλληλο πρόγραμμα για τον πολλαπλασιασμό τετραγωνικών πινάκων με χρήση OpenMP tasks. Η κάθε εργασία έπρεπε να είναι ο υπολογισμός ενός block του αποτελέσματος μεγέθους SxS, όπου το S έπρεπε να είναι παραμετροποιήσιμο και να δίνεται ως παράμετρος κατά την εκτέλεση του προγράμματος.

Υλοποίηση

Για την υλοποίηση της άσκησης δημιούργησα μία παράλληλη περιοχή, δήλωσα τον τύπο των συμμετεχόντων μεταβλητών και έθεσα το πλήθος των νημάτων ίσο με 4. Στη συνέχεια, όρισα μια περιοχή μέσα σ’ αυτήν, στην οποία ένα νήμα δημιουργεί ένα βρόγχο και ορίζει μία δουλειά που θα μπορεί να την αναλάβει αυτό ή κάποιο άλλο νήμα. Η δουλειά αυτή είναι ο υπολογισμός ενός block του τελικού πίνακα και βρίσκεται στη συνάρτηση taskexecute. Κάθε φορά που μία δουλειά τελειώνει ο μετρητής αυξάνεται και η διαδικασία συνεχίζεται μέχρι να ολοκληρωθούν όλες οι δουλειές.

Η συνάρτηση taskexecute παίρνει ως ορίσματα τον αριθμό της τρέχουσας δουλειάς, ώστε το νήμα να ξέρει το αντίστοιχο block που πρόκειται να υπολογίσει στον πίνακα, το S που είναι το μέγεθος του block και το M, που υπολογίζεται από το S και είναι το πλήθος των blocks.

Χρονομέτρηση

Για τη χρονομέτρηση χρησιμοποίησα σε όλες τις περιπτώσεις κλήσεις χρονομέτρησης που παρέχει το OpenMP *(omp_get_wtime())*.

Αρχικά, χρονομέτρησα το σειριακό πρόγραμμα. Στη συνέχεια, χρονομέτρησα το παράλληλο πρόγραμμα για 3 διαφορετικά πλήθη εργασιών 16, 256 και 1024.

Όλες οι χρονομετρήσεις φαίνονται στους παρακάτω πίνακες:

Σειριακό

1η εκτέλεση	**2η εκτέλεση**	**3η εκτέλεση**	**4η εκτέλεση**	**Μέσος Όρος**
3.997494	3.857529	3.846237	4.473169	4.04360725

Παράλληλο
S = 16

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.746021	0.746105	0.745991	0.746391	0.746127

Παράλληλο
S = 256

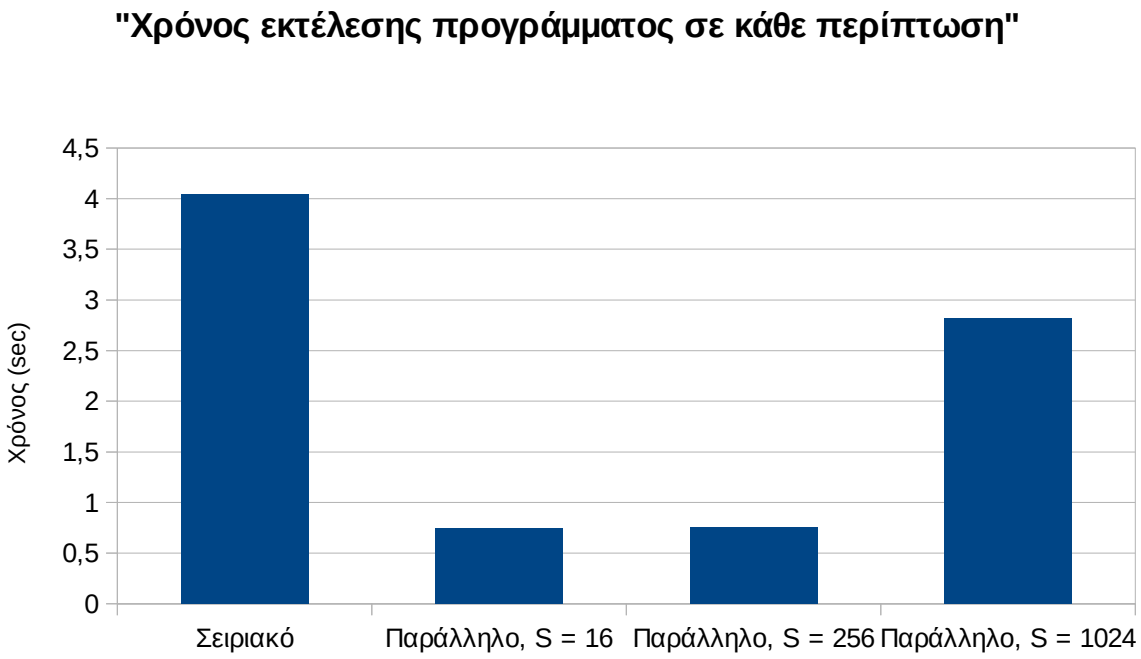
1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
0.757884	0.752742	0.753466	0.752797	0.75422225

Παράλληλο
S = 1024

1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	4η εκτέλεση	Μέσος Όρος
2.814707	2.815269	2.839134	2.824228	2.8233345

Γραφικές Παραστάσεις

Με βάση τους παραπάνω πίνακες δημιούργησα μία γραφική παράσταση που συγκρίνει το σειριακό πρόγραμμα με τα 3 διαφορετικά πλήθη εργασιών του παράλληλου προγράμματος.



Παρατηρήσεις

Παρατηρώ ότι σε κάθε περίπτωση το παράλληλο πρόγραμμα είναι πιο γρήγορο από το σειριακό. Ιδιαίτερα στην περίπτωση που το πλήθος των εργασιών ισούται με 16 και 256 παρουσιάζει το χαμηλότερο χρόνο με ελάχιστα πιο γρήγορο να είναι αυτό με τις 16 εργασίες. Στην περίπτωση των 1024 εργασιών είναι λογικό που ο χρόνος είναι αρκετά αυξημένος, αφού παρουσιάζεται το φαινόμενο του λεπτόκοκκου.