Τζέτζης Μάρκος ΑΜ:2142

Σετ προγραμμάτων #1

1)Υπολογισμός του π με νήματα

Με 1 νημα

#εργασιών(K)	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	18.265847	18.261293	18.266946	18.264417	18.264625
10	3.260376	3.265673	3.260226	3.261285	3.26189
100	2.045359	2.047959	2.044267	2.050303	2.046972
1000	1.920359	1.920310	1.917890	1.920262	1.919705
10000	1.906563	1.900797	1.910205	1.904075	1.90541
100000	1.904925	1.898237	1.919746	1.904681	1.90689725

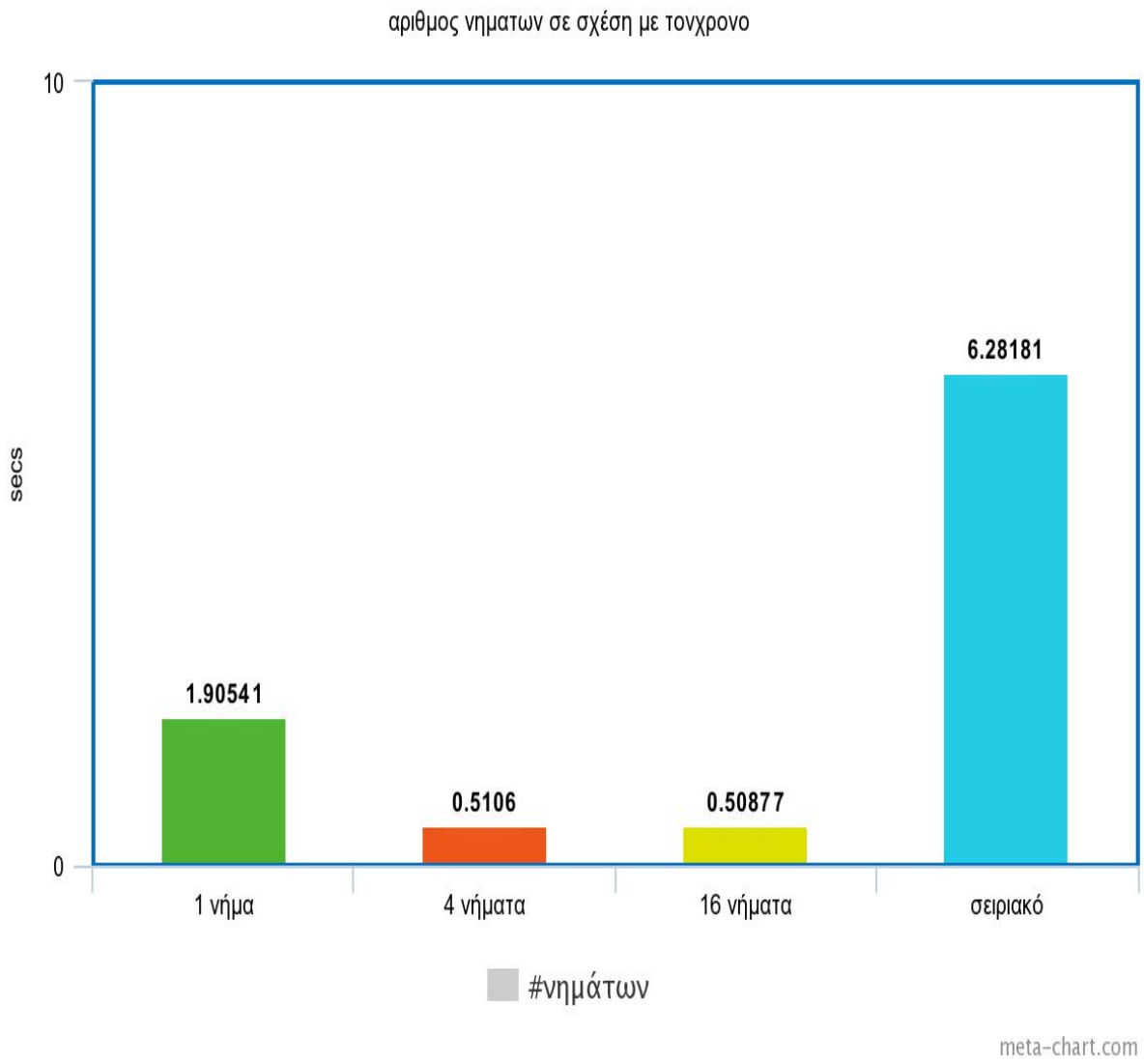
Με 4 νηματα

#εργασιών(K)	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	102.681491	98.112582	96.490689	97.110584	98.598836
10	11.778273	12.386018	11.835255	11.709514	11.927265
100	0.786803	0.775799	0.796387	0.786634	0.78640575
1000	0.518654	0.522833	0.520847	0.520933	0.52081685
10000	0.513753	0.511267	0.512214	0.511271	0.51212625
100000	0.513720	0.508809	0.509929	0.509973	0.51060775

Με 16 νηματα

#εργασιών(K)	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	92.782225	91.791966	90.284181	91.513647	91.59300475
10	11.419359	11.236876	11.305492	11.325223	11.3217375
100	0.783574	0.787357	0.776262	0.787754	0.78373675
1000	0.524263	0.522494	0.519864	0.524321	0.5227355
10000	0.511228	0.508319	0.509220	0.506134	0.50872525
100000	0.506185	0.509966	0.505350	0.509984	0.50787125

σειριακο	6.305100	6.287977	6.265330	6.268847	6.2818135



Παρατηρουμε οτι ο χρονος στην προσεγγιση με τα νηματα βελτιωνεται αισθητα.Οι περιπτωσεις που φαινονται στο γραφημα αναφεροντε στα βελτιστα K .Ενδιαφερον παρουσιαζει το γεγονος οτι οταν το K γινεται μεγαλυτερο απο 1000 παρουσιαζεται μικρη βελτιωση στην αποδοση κατι που ειναι λογικο καθως οσο μεγαλυτερο το K τοσο πιο λεπτοκοκκος ο παραλληλος αλγοριθμος,εν αντιθεση οσο πιο μικρο το K τοσο πιο αργο απο το σειριακο(πχ για K=10) στο συγκεκριμενο προγραμμα φαινεται πως η “χρυση” τομη ειναι το K=1000.Επισης παρατηρουμε οτι ο βελτιστος χρονος για αριθμο νηματων 4 και 16 δν διαφερει αισθητα κατι που οφειλεται στο γεγονος οτι το μηχανημα που ετρεξαν τα αρχεια(opti7020ws09) διαθετει 4 cores που σημαινει οτι τα παραπανω απο 4 νηματα δεν προσδιδουν σμαντικη βελτιωση στην αποδοση.Εαν ο αλγοριθμος δν ετρεχε με αυτοδρομολογηση θεωρω οτι ο χρονος που θα βλεπαμε στα 16 νηματα θα ηταν ελαφρα χειροτερος.Αξιοσημειωτο ειναι και το γεγονος οτι ακομα και με 1 νημα για K > 1 ο χρονος ειναι καλυτερος απο τον αντιστοιχο σειριακο , αυτο γινεται επειδη τα νηματα εινια πιο ελαφρια απο τις διεργασιες .

2)Πολλαπλασιασμός πινάκων με νήματα

256

#νημάτων	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	0.043710	0.043768	0.045504	0.045091	0.04451825
2	0.025068	0.024631	0.023292	0.024151	0.0242855
4	0.020496	0.015400	0.017375	0.018176	0.01786175
8	0.017880	0.016111	0.017540	0.015477	0.016752
12	0.014805	0.020489	0.015214	0.018334	0.0172105
16	0.014124	0.015336	0.015807	0.014324	0.01489775

σειριακό	0.046177	0.045872	0.044747	0.042989	0.04494625

512

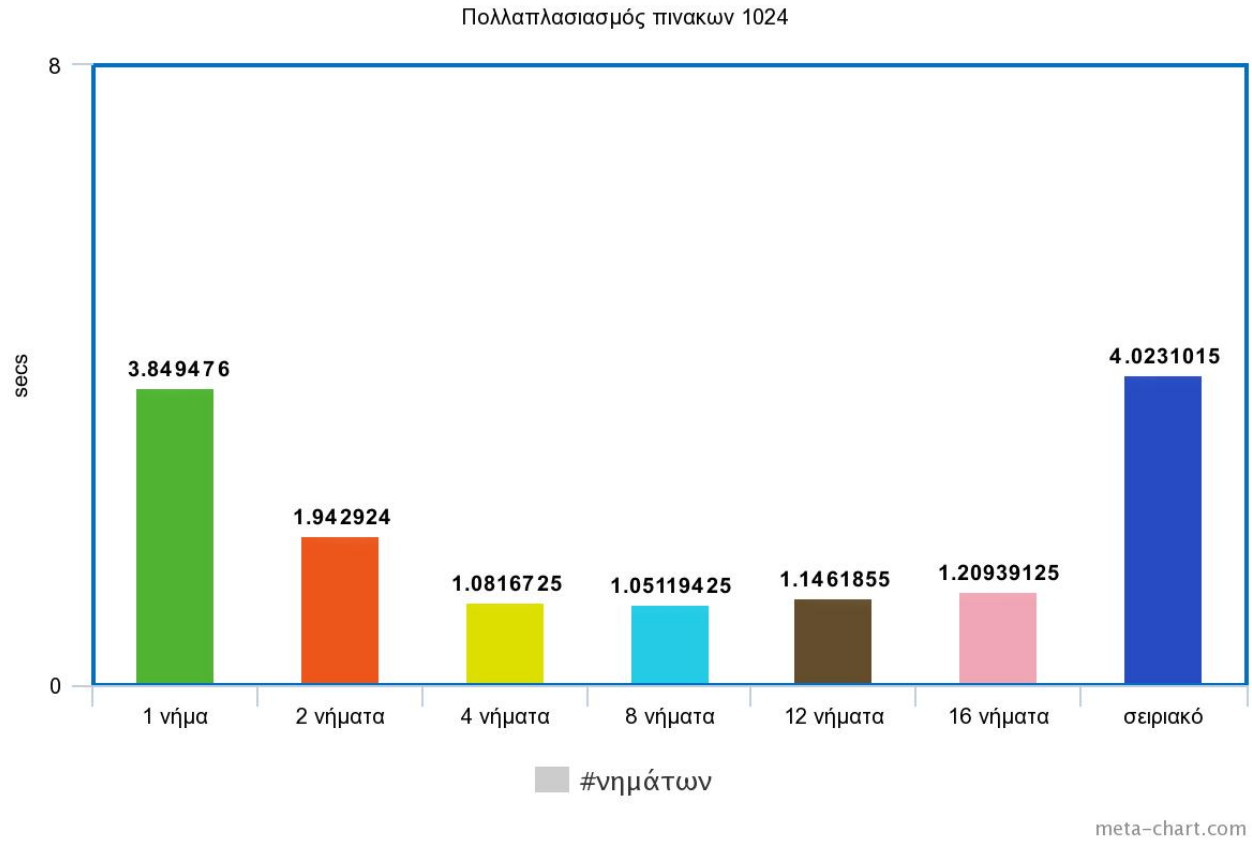
#νημάτων	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	0.376163	0.378870	0.378706	0.379281	0.378255
2	0.191208	0.194751	0.198536	0.190319	0.1937035
4	0.104095	0.106011	0.105004	0.104410	0.10488
8	0.104929	0.112090	0.112115	0.108290	0.109356
12	0.107739	0.107228	0.108711	0.103048	0.1066815
16	0.106494	0.105741	0.103203	0.105763	0.10530025

σειριακό	0.376333	0.377327	0.376812	0.377465	0.37698425

1024

#νημάτων	1 run	2 run	3 run	4 run	Μέσος Χρόνος
1	2.984610	4.806339	3.766760	3.840207	3.849479
2	2.254065	1.915721	1.609380	1.992530	1.942924
4	1.136747	1.010071	0.943925	1.235947	1.0816725
8	1.012946	0.986106	1.021122	1.184603	1.05119425
12	1.326071	0.866846	1.288357	1.103468	1.1461855
16	1.187312	1.316337	1.206588	1.127328	1.20939125

σειριακό	4.610546	3.881896	4.540790	3.048014	4.0203115



Στο δευτερο προγραμμα φαινεται πιο ξεκαθαρο το γεγονος οτι το προγραμμας μας συμπεριφερεται καλυτερα για αριθμο νηματων 4-8 επειδη αξιοποιουνται ολα τα cores ,Παρατηρουμε οτι για μεγαλυτερο αριθμο νηματων η αποδοση ειναι χειροτερη κατι που ειναι λογικο γιατι ξοδευουμε αρκετο χρονο στην δημιουργια των νηματων.Το προγραμμα ειναι γραμμενο χωρις κλειδαριες επειδη σε αυτην την υλοποιηση το κομματι των πολλαπλασιασμων του πινακα που εκτελει το καθε νημα ειναι ανεξαρτητο απο τα αλλα νηματα.Καθε νημα εκτελει **RPT N/NTHR** επαναλήψεις οπου N το μεγεθος του πινακα και NTHR ο αριθμος των νηματων.Η παραλληλοποιηση γινεται στον εξωτερικο βρογχο γραμμικα κατι που μπορει να γινει σε αυτην την περιπτωσει καθως το NTHR διαιρει ακριβως το N καθως και τα 2 ειναι πολλαπλασια του 2.Σε αντιθετη περιπτωση καποιο νημα θα εκαν λιγοτερη δουλεια απο τα άλλα .Δοκιμάστικε και η παραλληλοποίηση του εξωτερικου βρογχου με αλματα αλλα η προσεγγίση αυτη αποριφθηκε καθως ηταν πιο αργη ,γεγονος που οφειλεται στην χρονικη αλλα κυριως στην χωρικη τοπικοτητα που εχουν οι cache.Τα διαγραμματα για 256 και 512 ειναι αντιστοιχα.Τελος ολες οι μετρησεις εγινα στον opti7020ws09.

3)Υλοποίηση απλού barrier

Η υλοποιηση του barrier εγινε ως εξης: δημιουργησα ενα struct με μεταβλητες το remain που θα κρατουσε των αριθμο των νηματων που θελουμε να περιμενουν, μια μεταβλητη συνθηκης στην οποια μπλοκαρουν τα νηματα μια μεταβλητη mutex η οποια προστατευει την κρισιμη περιοχη που στην περιπτωση μας ειναι η μειωση του remain και ο ελεγχος των συνθηκων της barrier_wait.Ο barrier δουλεύει σωστα την πρωτη φορα αλλα την δευτερη φορα κολλαει.Δοκιμασα με το που γινεται το broadcast και ξυπνανε ολα τα νηματα που περιμενουν να δινεται στο remain η αρχικη του τιμη η οποια ειναι κρατημενη σε μια μεταβλητη count του barrier αλλα παρουσιαστηκε το προβλημα οτι μονο το πρωτο νημα εβγαινε απο το wait και τα αλλα λογο της αλλαγης της τιμης παρεμεναν μπλοκαρισμενα.Σκεφτηκα να αλλαζω την τιμη σε εναν bool flag για να γνωριζει το wait οτι αυτη ειναι η επομενη φορα που καλειται αλλα το προβλημα παρεμεινε το ιδιο.Για την υλοποιηση του init αρχικοποιησα την κλειδαρια και την μεταβλητη συνθηκης καθως και τα counter,flag,remain.Για το destroy κρατησα ενα αρχικο αντιγραφω του struct του barrier και με την κληση του destroy αναθετω στον struct αυτο το αθικτο αντιγραφω.Τελος εγιναν διαφορα τεστ ταχυτητας με τον ετοιμο barrier ο οποιος ηταν και πιο γρηγορος αλλα η διαφορα δεν ηταν τοσο μεγαλη οσο περιμενα.