Project 3:
Παραλληλισμός με χρήση OpenMP

Εφαρμότηκε παραλληλισμός στις nested for loops που αφορούσαν τη σύγκριση των αλληλουχιών των δύο σετ (mset & nset).

```
for m
   for n
      for l
```

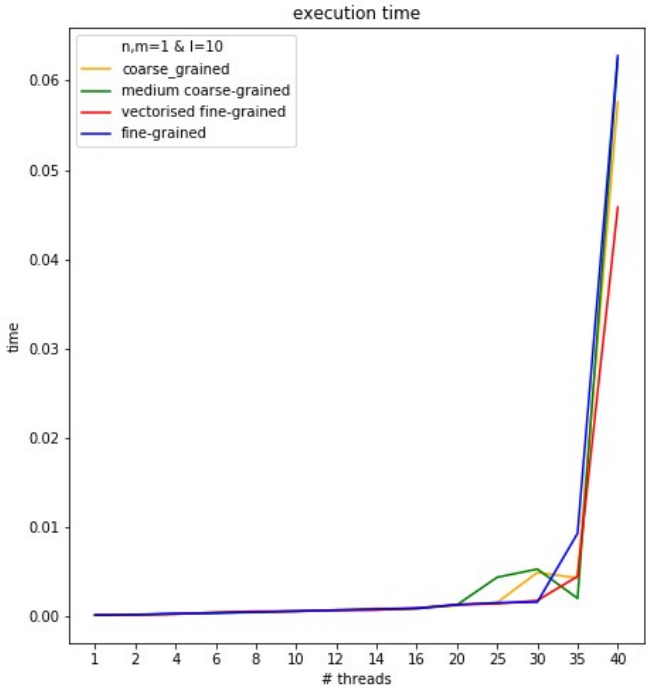
Ως fine-grained ορίστηκε ο παραλληλισμός στην πιο εσωτερική for loop (for l), η ανάθεση δηλαδή σε κάθε thread μικρών tasks, ίσα με l/number of threads κάθε φορά. Αυτό σημαίνει ότι κάθε thread έλεγχε μία θέση από κάθε αλληλουχία.
Ως medium coarse-grained ο παραλληλισμός στην ενδιάμεση for loop (for n), η ανάθεση μεγαλύτερων tasks, μεγέθους ίσου με n/number of threads. Σε αυτή την περίπτωση, κάθε thread σύγκρινε μία αλληλουχία του mset με μία του nset gia όλες τις θέσεις l κάθε αλληλουχίας.
Τέλος, ως coarse-grained ο παραλληλισμός στην εξωτερική for loop (for m), όπου κάθε thread αναλάμβανε για κάθε μία αλληλουχία του mset να τη συγκρίνει με όλες του nset και για όλες τις θέσεις l αυτών.

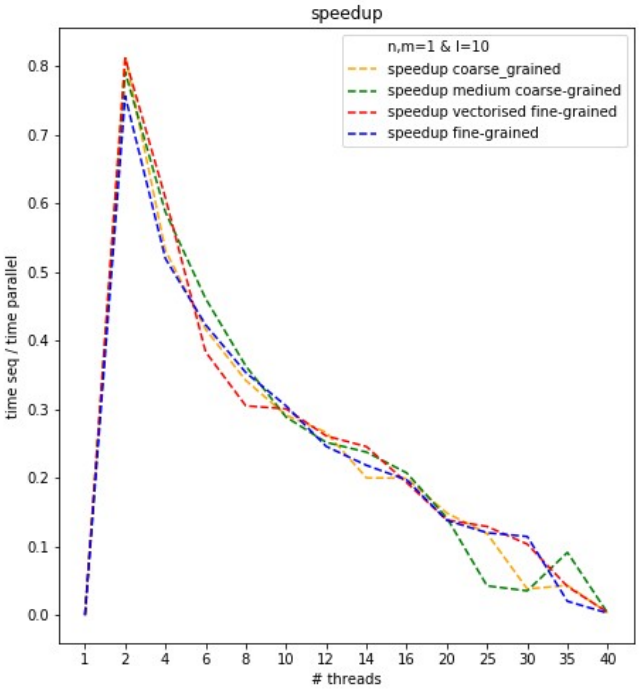
Η απόσταση των αλληλουχιών καταγραφόταν σε ένα πίνακα mn μεγέθους m*n.
Κατά την εκτέλεση ο fine-grained παραλληλισμός έδινε λάθος άθροισμα, ως αποτέλεσμα του false sharing, μιας και κάθε thread αύξανε την απόσταση κατά ένα, αλλά άλλαζαν όλα την ίδια μεταβλητή d, οπότε και δεν διάβαζαν όλα τη σωστή τιμή d. Το ίδιο συμβαίνει και κατά την άθροιση των αποστάσεων και την ανάγνωση και εγγραφή του checksum, της μεταβλητής που αποθηκεύει το άθροισμα αυτών των αποστάσεων. Επιπλέον, η πρόσβαση σε στοιχεία που έχουν εγγραφεί σε κοντινές θέσεις στη μνήμη από διαφορετικά threads, προκαλεί και καθυστέρηση της εκτέλεσης του προγράμματος, μιας και πρέπει να συγχρονιστούν αυτά τα threads μεταξύ τους.

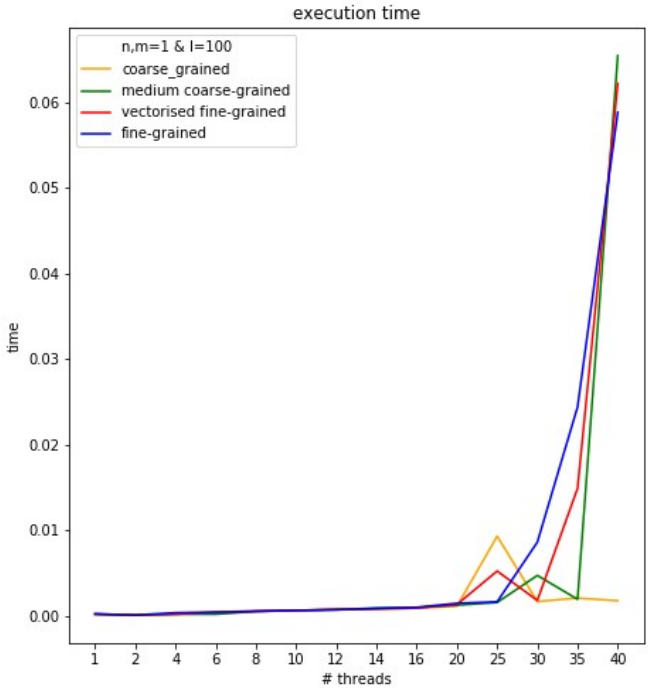
Ως προσπάθεια επίλυσης του false sharing, τα δεδομένα των πινάκων mset, nset & mn εγγράφηκαν στη μνήμη ευθυγραμμισμένα σε εύρος 64 bytes, όσο είναι και το μήκος μιας cache line, χρησιμοποιώντας την _mm _malloc. Στην parallel for loop τα δεδομένα διαβάζονταν ανά 16 (16*sizeof(int)==64 bytes) για τη σύγκριση των θέσεων των αλληλουχιών και τον υπολογισμό της απόστασης, ενώ για την άθροιση των αποστάσεων το βήμα ήταν ανά 8 (8*sizeof(unsigned long long) == 64 bytes) και κάθε thread αποθήκευε την δική του τιμή d ή checksum, ωστε να αποφευχθεί η ανανέωση των ίδιων μεταβλητών από διαφορετικά threads. Αυτή η υλοποίηση είναι η vectorized fine-grained. Με αυτόν τον τρόπο, το άθροισμα της απόστασης εμφανιζόταν ίσο με τις υπόλοιπες υλοποιήσεις, ο χρόνος όμως εκτέλεσης δεν βελτιώθηκε σε σχέση με την fine-grained υλοποίηση.
Ο χρόνος εκτέλεσης αυξήθηκε δραματικά για n,m=10000 και για όλα τα διαφορετικά l, όσο αυξανόταν ο αριθμός των threads γι’ αυτό και στα διαγράμματα που ακολουθούν τα threads για n,m=10000 & l=10 σταματούν στα 35 και στα n,m = 10000 & l=100 & l=1000 η σύγκριση σταματάει στα 20 threads.

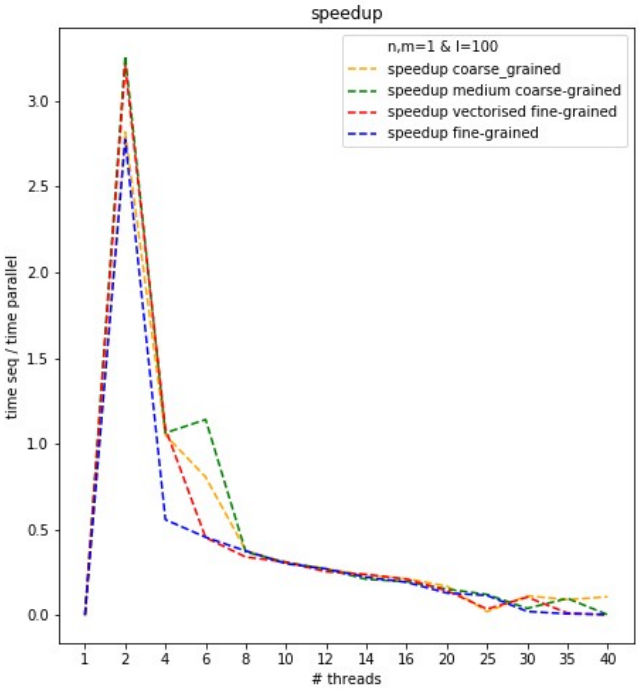
Για n,m = 1 και ανεξαρτήτως του μήκους των αλληλουχιών, όλες οι υλοποιήσεις επιτυγχάνουν περίπου τους ίδιους χρόνους εκτέλεσης και άρα και επιτάχυνσης και όσο αυξάνεται ο αριθμός των threads η επιτάχυσνη μειώνεται.
Για n,m = 100 και πάνω οι υλοποιήσεις fine-grained χρειάζονται πολύ περισσότερο χρόνο ακόμη και από τον σειριακό κώδικα. Η coarse-grained υλοποίηση τρέχει γρηγορότερα από την medium coarse-grained σε γενικές γραμμές, όμως σε μεγαλύτερες τιμές των n,m (>1000) και l (>100) η απόδοσή τους

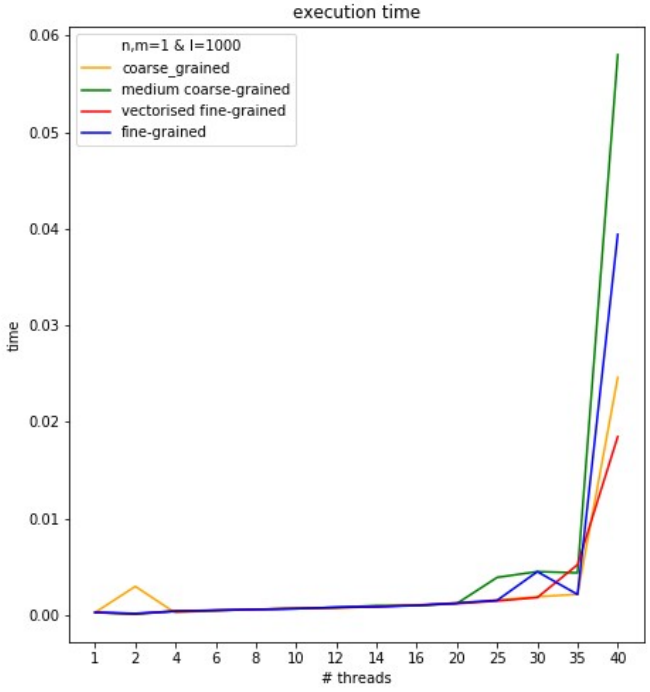
κυμαίνεται στα ίδια επίπεδα. Για n,m =100 και όσο αυξάνεται το μήκος των αλληλουχιών παρατηρείται ότι το κατώφλι για τον αριθμό των threads που επηρεάζει την απόδοση μετακινείται από τα 2-6 στα 25-30. Για μεγαλύτερες τιμές των n,m και όσο αυξάνεται το l, η αύξηση του αριθμού των threads επιταχύνει τους υπολογισμούς.

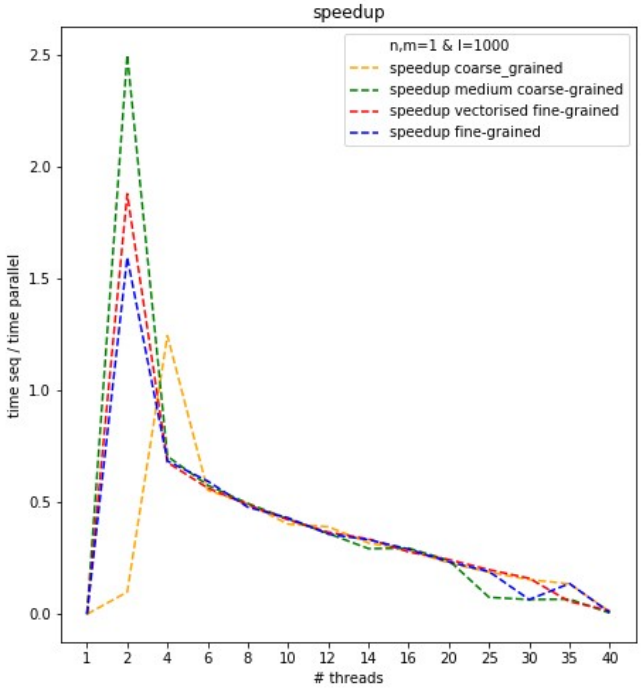


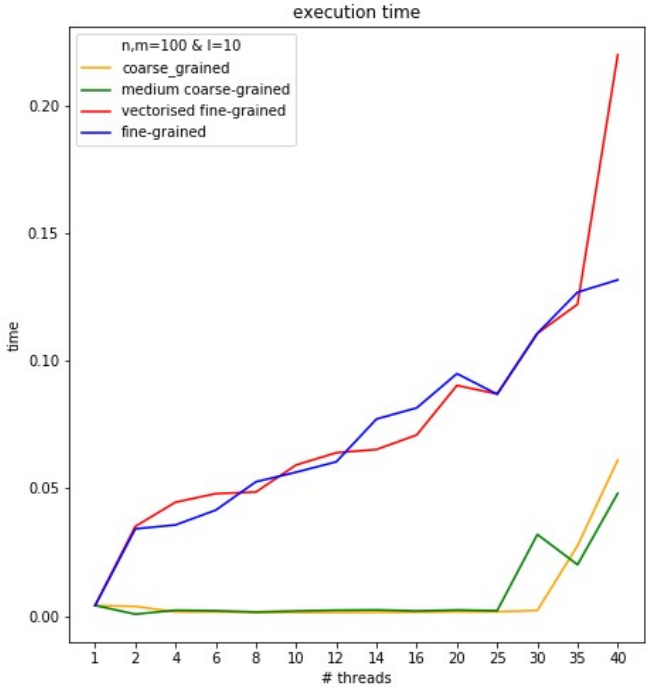


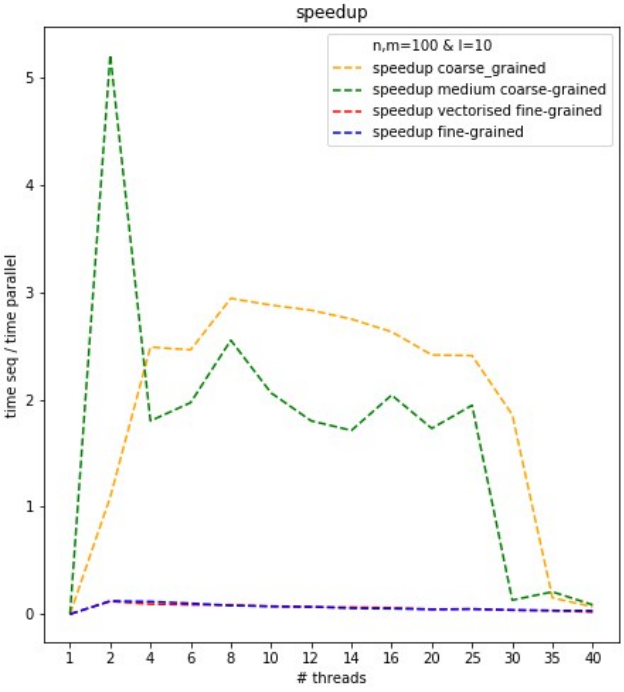


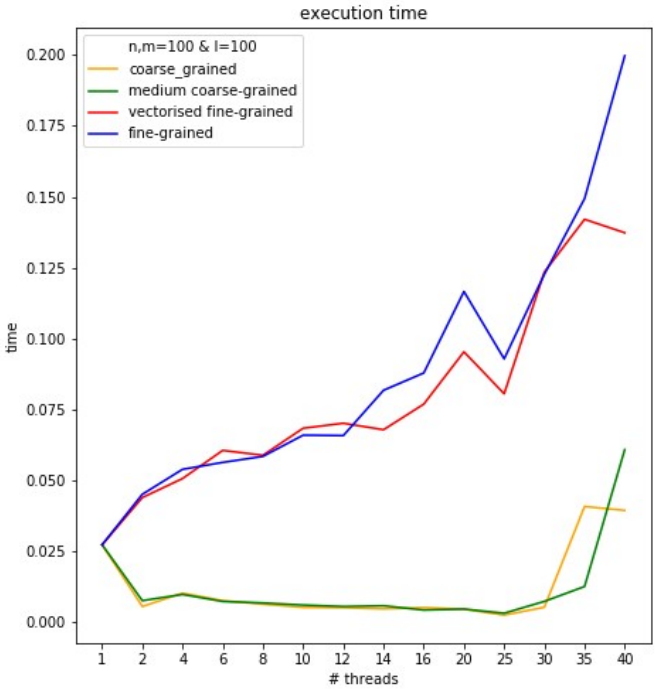


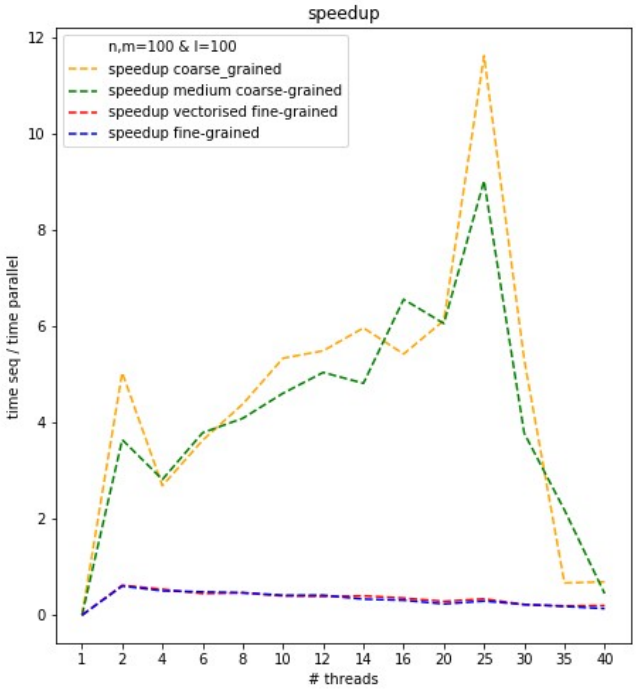

execution time
n,m=1 & l=1000
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

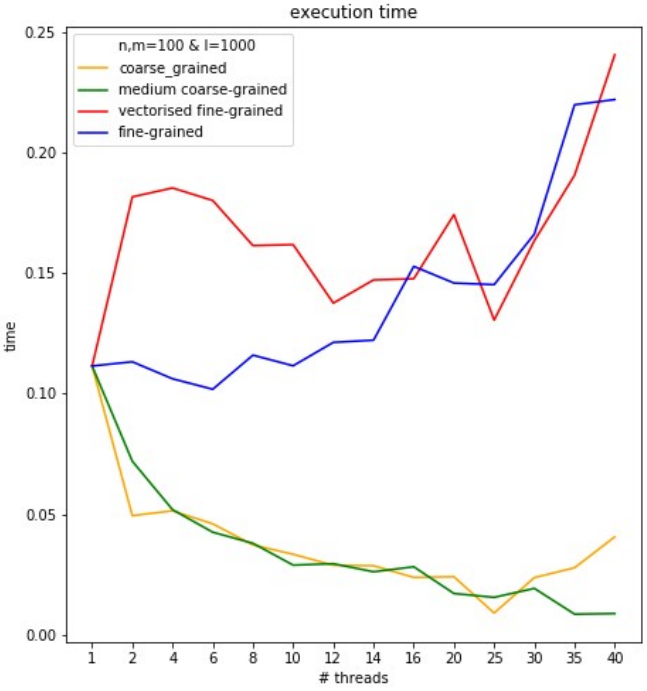

speedup
n,m=1 & l=1000
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

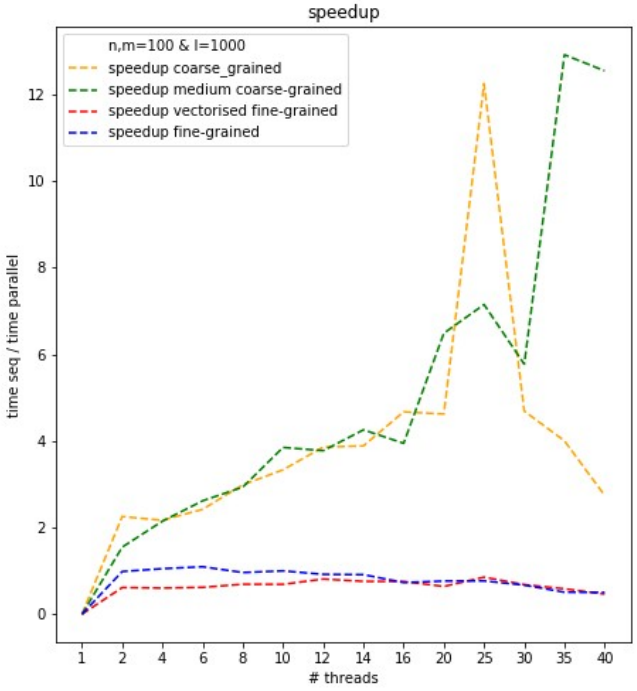

execution time
n,m=100 & l=10
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

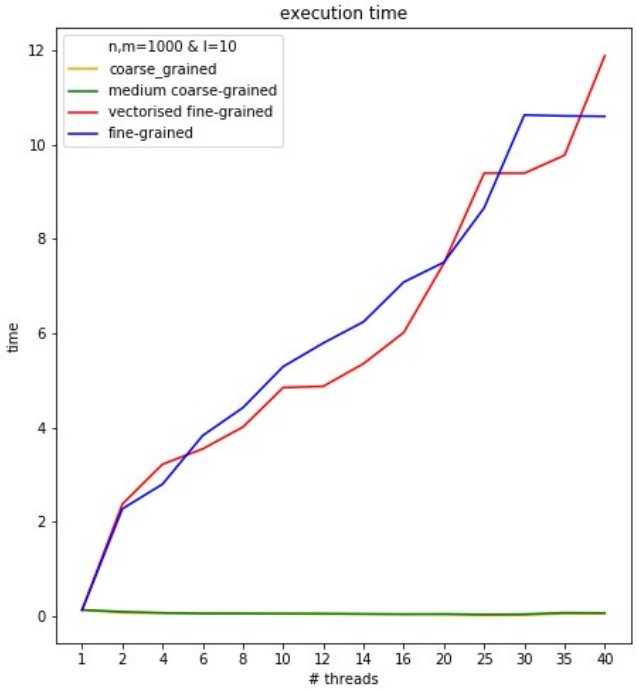

speedup
n,m=100 & l=10
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

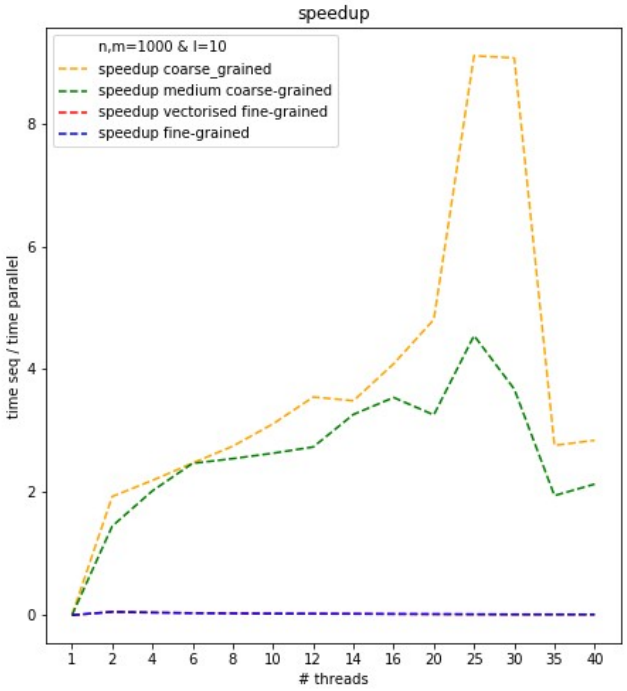

execution time
n,m=100 & l=100
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

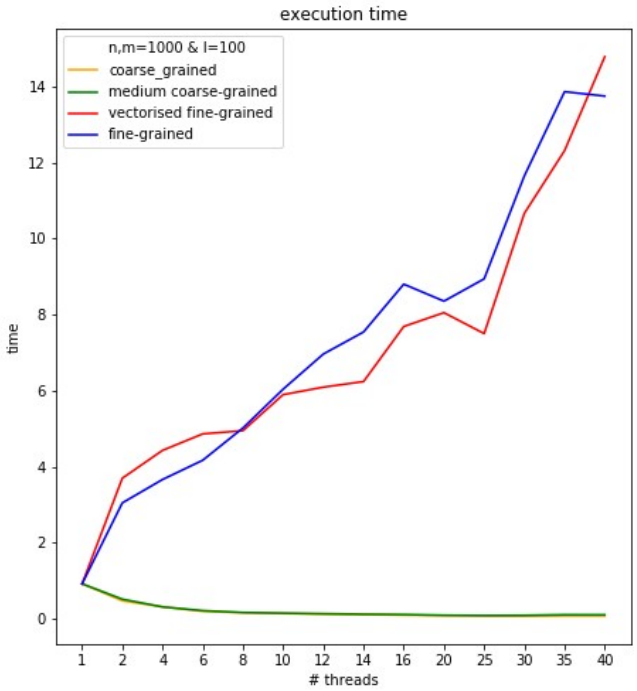

speedup
n,m=100 & l=100
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

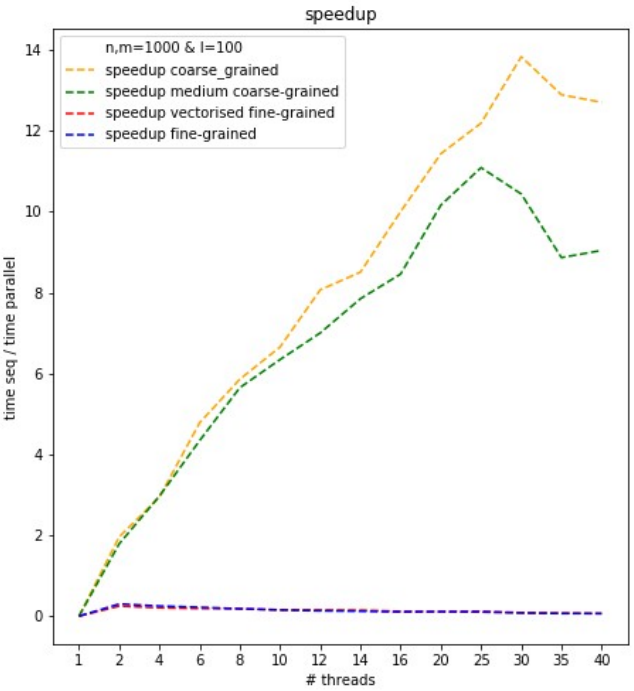

execution time
n,m=100 & l=1000
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

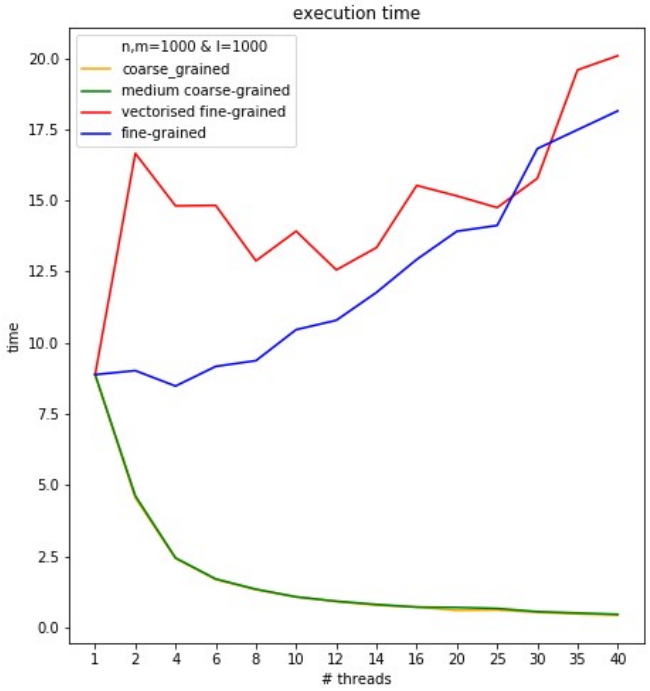

speedup
n,m=100 & l=1000
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

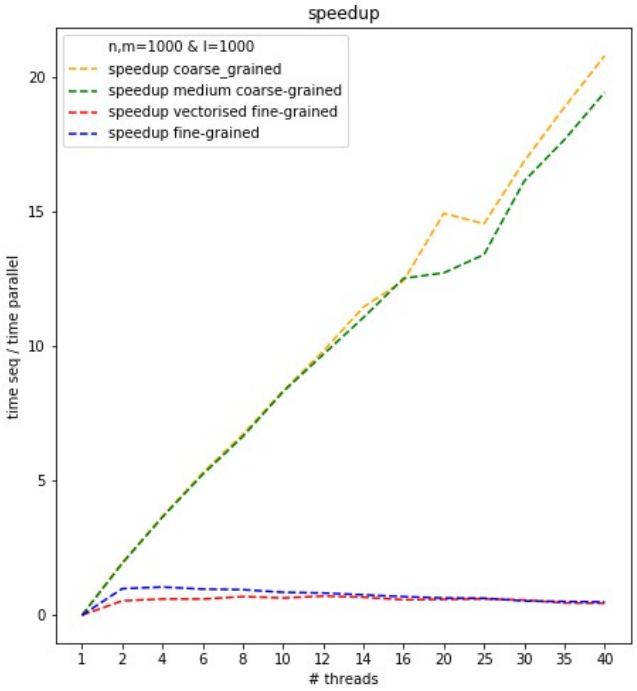

execution time
n,m=1000 & l=10
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

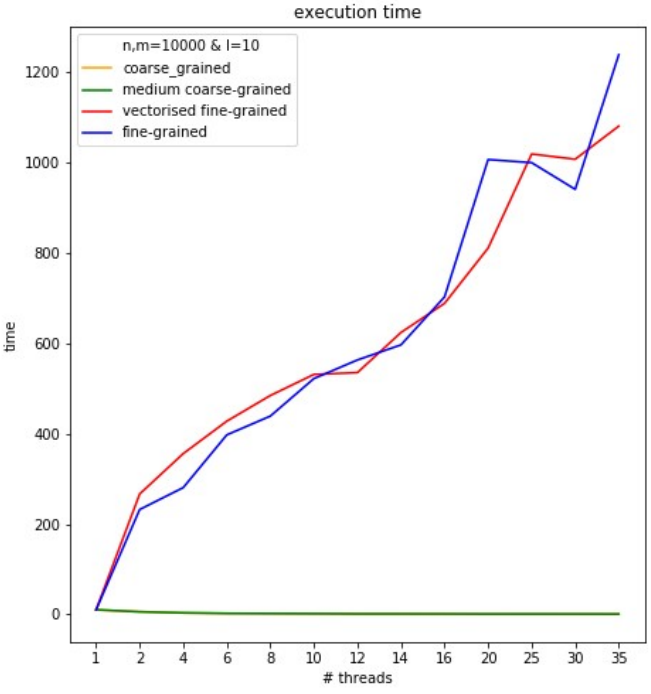

speedup
n,m=1000 & l=10
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

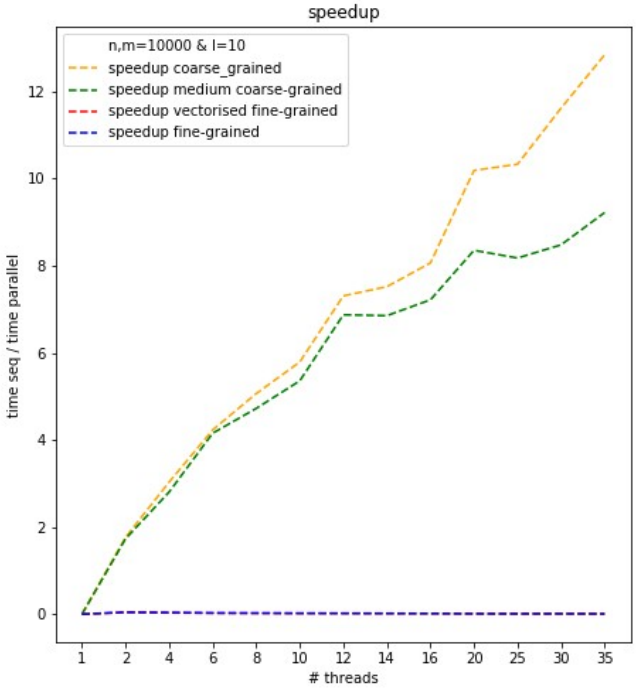

execution time
n,m=1000 & l=100
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

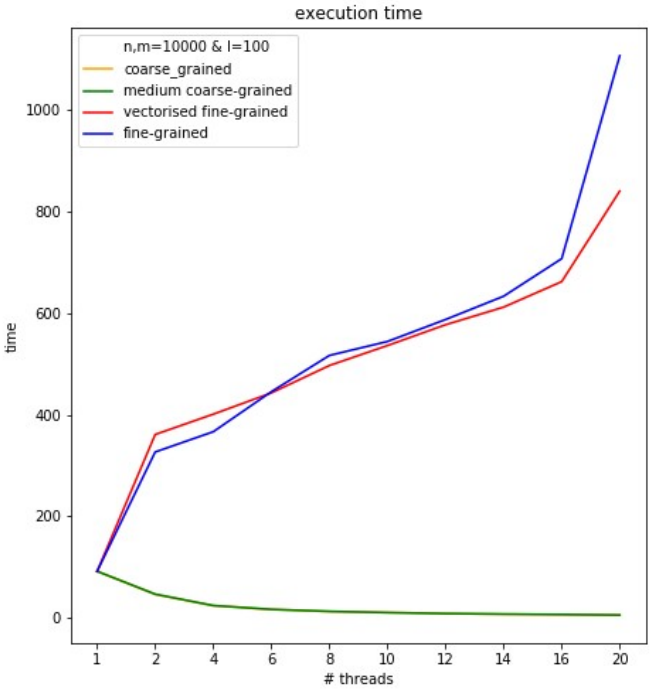

speedup
n,m=1000 & l=100
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

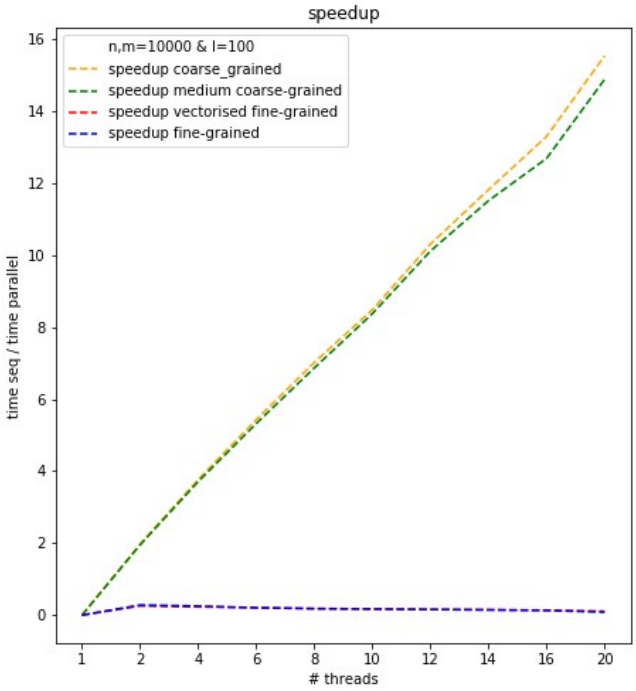

execution time
n,m=1000 & l=1000
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads

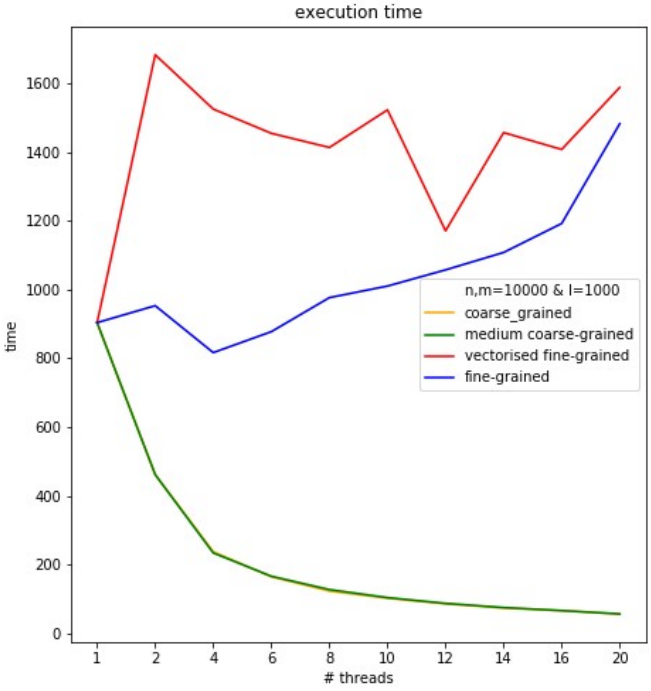

speedup
n,m=1000 & l=1000
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

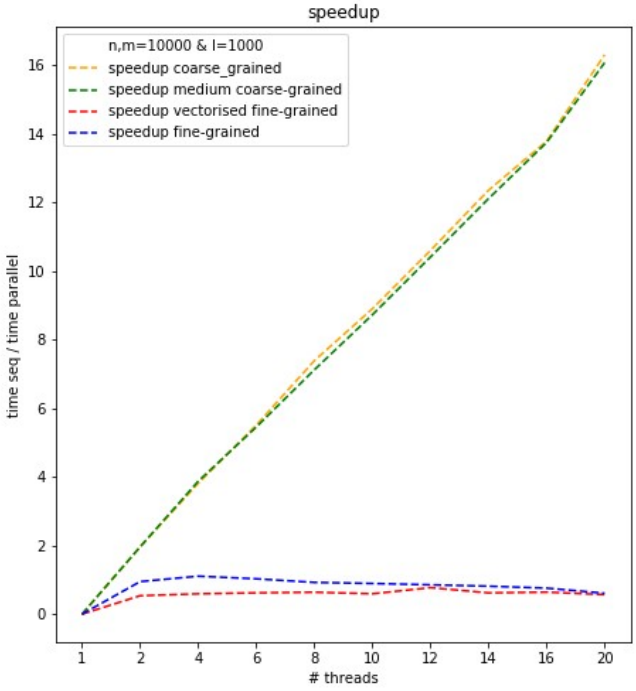

execution time
n,m=10000 & l=10
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads


speedup
n,m=10000 & l=10
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads


execution time
n,m=10000 & l=100
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads


speedup
n,m=10000 & l=100
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads


execution time
n,m=10000 & l=1000
coarse_grained
medium coarse-grained
vectorised fine-grained
fine-grained
time
threads


speedup
n,m=10000 & l=1000
speedup coarse_grained
speedup medium coarse-grained
speedup vectorised fine-grained
speedup fine-grained
time seq / time parallel
threads

Ο κώδικας έτρεξε σε σύστημα με τα εξής χαρακτηριστικά:

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              40
On-line CPU(s) list: 0-39
Thread(s) per core:  2
Core(s) per socket:  10
Socket(s):           2
NUMA node(s):        2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               85
Model name:          Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz
Stepping:            4
CPU MHz:             800.388
CPU max MHz:         3000.0000
CPU min MHz:         800.0000
BogoMIPS:            4400.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            1024K
L3 cache:            14080K
NUMA node0 CPU(s):   0-9,20-29
NUMA node1 CPU(s):   10-19,30-39
```