

**ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ
ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2013/2014**

ΟΝΟΜΑΤΑ: Πατρώνη Σωτηρία	ΑΜ:5399
Μιλκάι Έλενα	5356
Γώγουλου Ευαγγελία	5284

Ακολουθούν τα χαρακτηριστικά του υπολογιστικού συστήματος στο οποίο εκτελέστηκε τόσο η σειριακή όσο και η παράλληλη εφαρμογή:

Ο τύπος επεξεργαστή είναι Intel(R) Core(TM) i5 CPU M 480 και η συχνότητα λειτουργίας του είναι 2.67GHz

Αριθμός πυρήνων=2

enabled cores=2

threads=4

Το μέγεθος της κρυφής μνήμης και ο αριθμός επιπέδων της φαίνονται αναλυτικά παρακάτω:

- Cache L1

size: 32KiB

δυνατότητες: synchronous internal write-through instruction

- Cache L2

size: 256KiB

δυνατότητες: synchronous internal write-through unified

- Cache L3

size: 3MiB

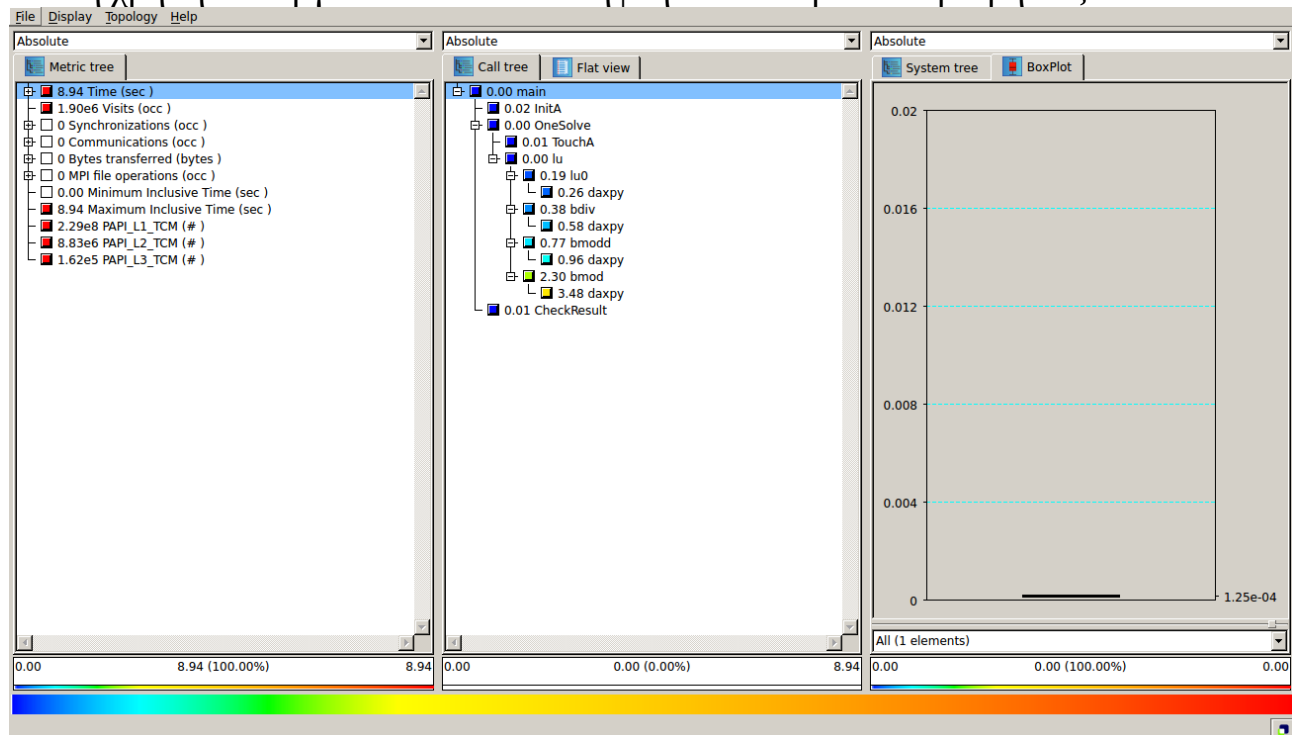
δυνατότητες: synchronous internal write-through unified

Ερώτημα 1ο:

Στην σειριακή έκδοση της παραγοντοποίησης LU κατά μπλοκ η βασική συνάρτηση που εκτελεί την παραγοντοποίηση του αρχικού πίνακα είναι η lu . Το μητρώο A χωρίζεται σε μπλοκ μεγέθους $bs \times bs$. Η παραγοντοποίηση εκτελείται επαναληπτικά στην lu (εξωτερική for) με την εξής σειρά: πρώτα παραγοντοποίηση του διαγώνιου μπλοκ, στην συνέχεια παραγοντοποίηση όλων των μπλοκ που βρίσκονται στην ίδια σειρά με το διαγώνιο μπλοκ, έπειτα παραγοντοποίηση σε όλα τα μπλοκ που βρίσκονται στην ίδια στήλη με το διαγώνιο και τέλος η ανανέωση όλων των υπολοίπων μπλοκ. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε στο τελικό διαγώνιο μπλοκ. Η συνάρτηση στην lu που αναλαμβάνει να παραγοντοποιήσει τα διαγώνια μπλοκ είναι η $lu0$. Η $lu0$ αποθηκεύει κατευθείαν στο κάτω τριγωνικό κομμάτι του κάθε διαγωνίου μπλοκ τους πολλαπλασιαστές που απαρτίζουν τις τιμές του L . Επίσης εκτελεί τις κατάλληλες γραμμοπράξεις με την χρήση των πολλαπλασιαστών που υπολόγισε και της $daxpy$, η οποία αφαιρεί πολλαπλάσιο μιας γραμμής από μια άλλη, και έτσι δημιουργείται το ανω τριγωνικό U του κάθε διαγωνίου μπλοκ. Η επόμενη συνάρτηση που εκτελείται στην lu είναι η $bdiv$ η οποία αναλαμβάνει με βάση τους πολλαπλάσιαστές που υπολόγισε στο προηγούμενο η $lu0$ βήμα να παραγοντοποιήσει όλα τα μπλοκ που ανήκουν στην ίδια γραμμή με το

αντίστοιχο διαγώνιο που παραγοντοποιήσε στην ίδια επανάληψη. Η παραγοντοποίηση των μπλοκ της ίδιας γραμμής γίνεται με την χρήση μιας for καθώς η διαδικασία είναι η ίδια κάθε φορά. Μετά την παραγοντοποίηση των μπλοκ της ίδιας γραμμής τα στοιχεία που θα περιέχονται στα μπλοκ αυτά θα είναι τα στοιχεία του άνω τριγωνικού U. Αμέσως μετά μια άλλη for αναλαμβάνει την παραγοντοποίηση των μπλοκ που ανήκουν στην ίδια στήλη με το αντίστοιχο διαγώνιο. Η συνάρτηση που εκτελεί αυτή την διαδικασία είναι η `bmodd`. Τα μπλοκ της ίδιας στήλης μετά την παραγοντοποίηση θα περιέχουν αποθηκευμένους μόνο πολλαπλασιαστές καθώς για αυτά τα κομμάτια ανήκουν στο κάτω τριγωνικό κομμάτι της U που περιέχει μόνο μηδενικά. Τέλος για να ολοκληρωθεί η παραγοντοποίηση στην μια στήλη και γραμμή μπλοκ θα πρέπει να ανανεωθούν και τα μπλοκ που δεν ανήκουν σε αυτά. Την διαδικασία την αναλαμβάνει η διπλή for στο τέλος της `lu`. Η συνάρτηση `bmod` με την `daxpy` αναλαμβάνουν να εκτελέσουν τις κατάλληλες γραμμοπράξεις με την χρήση των πολλαπλασιαστών που έχουν ήδη υπολογιστεί. Η σειρά με την οποία ανανεώνονται τα εσωτερικά μπλοκ είναι η εξής : αρχικά τα μπλοκ που βρίσκονται στην ίδια γραμμή και αμέσως μετά αυτά που βρίσκονται στην ίδια στήλη μέχρι οι γραμμές και οι στήλες να τελειώσουν μπαίνοντας κάθε φορά κατά ένα μπλοκ προς τα μέσα.

Με τη χρήση του εργαλείου Scalasca λήφθηκαν οι παρακάτω μετρήσεις:



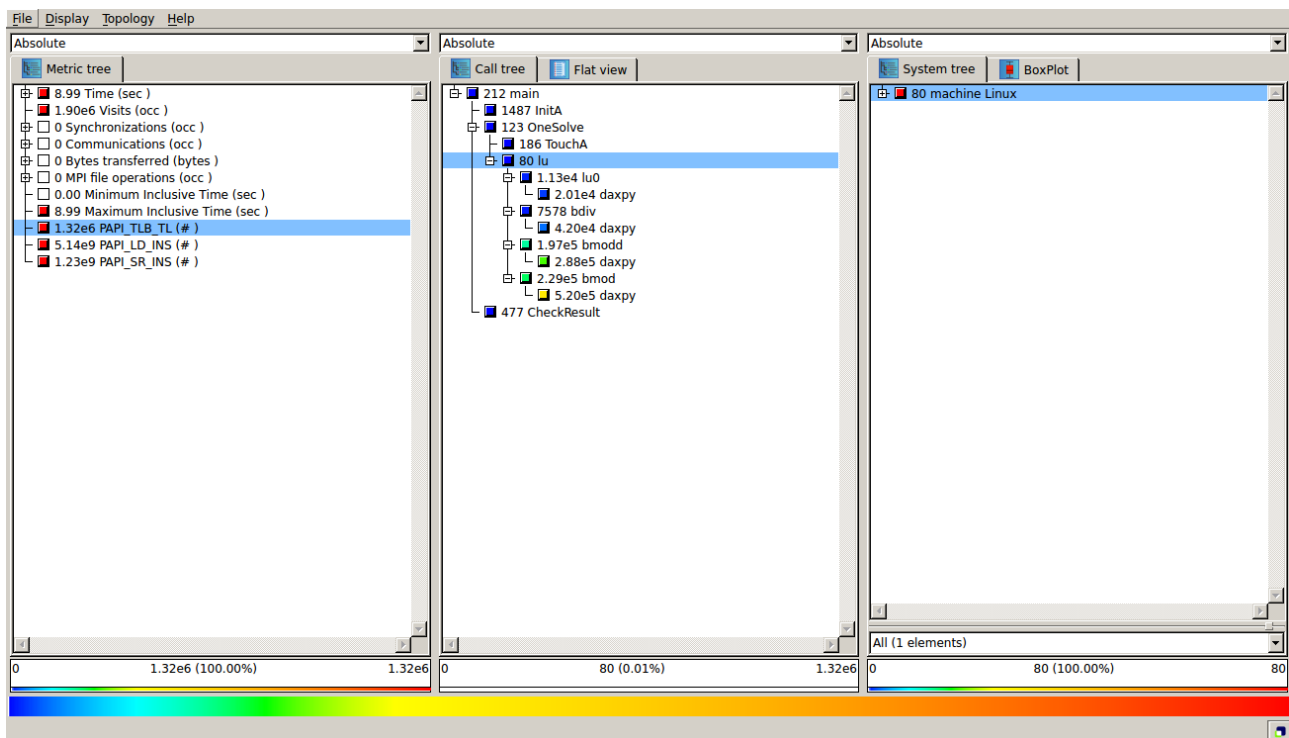
Όπως φαίνεται παραπάνω, η εκτέλεση της συνάρτησης `daxpy` που καλείται από τη `bmod` καθώς και η ίδια η `bmod` καταλώνει τον περισσότερο χρόνο σε σχέση με τις υπόλοιπες συναρτήσεις. Αυτό οφείλεται στο γεγονός ότι λόγω της `bmodd` στη cache βρίσκονται δεδομένα τα οποία δε προσπελαίνει αρχικά η συνάρτηση `daxpy`, η οποία καλείται από τη `bmod`. Συνεπώς προκαλούνται αρκετά cache misses τα οποία καθυστερούν την εκτέλεση της συνάρτησης `daxpy` καθώς και της `bmod`.

Οι μετρητές οι οποίοι επιλέχθηκαν είναι οι εξής:

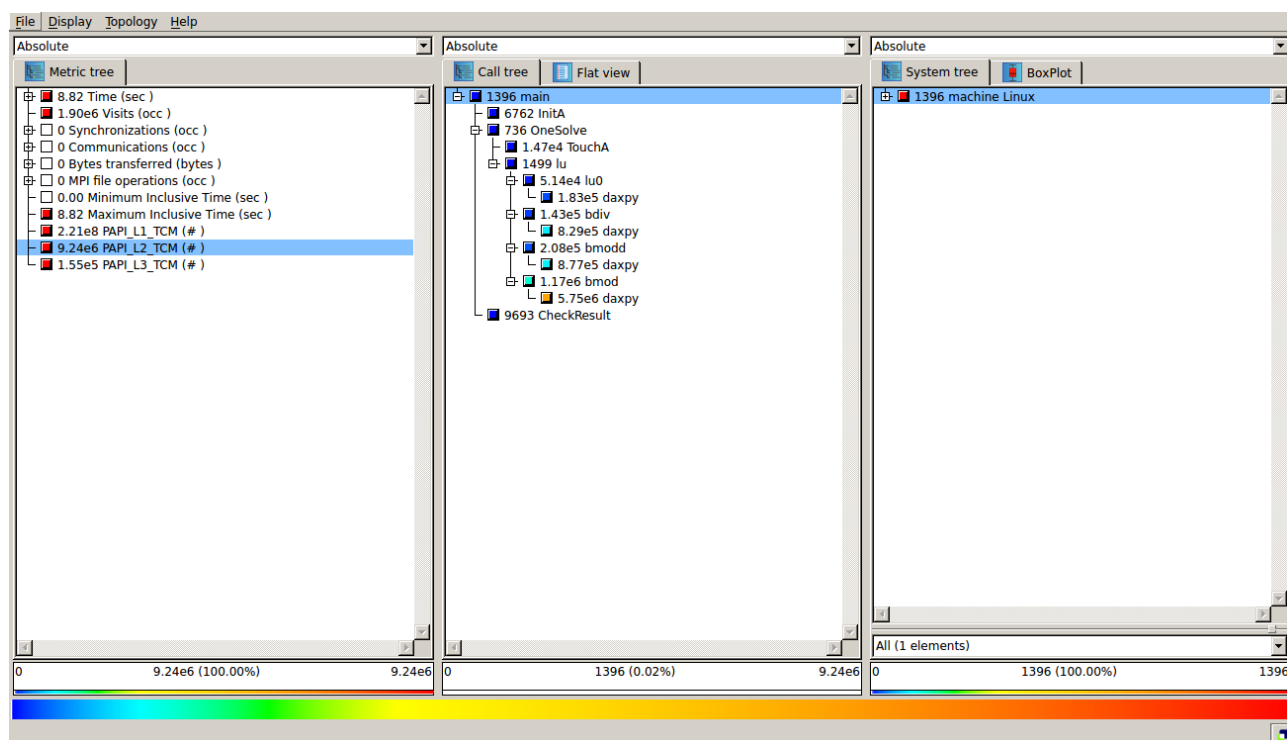
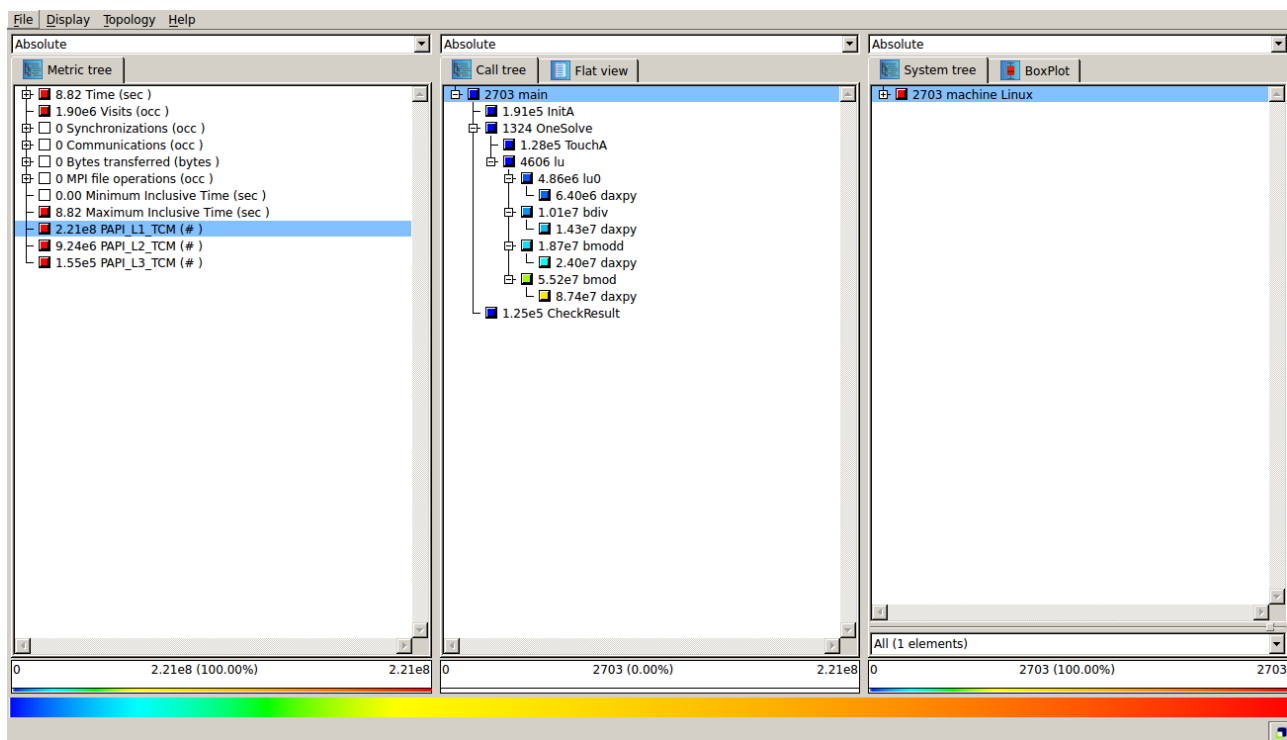
PAPI_L1_TCM	Level 1 cache misses
PAPI_L2_TCM	Level 2 cache misses
PAPI_L3_TCM	Level 3 cache misses
PAPI_TLB_TL	Total translation lookaside buffer misses
PAPI_FP_INS	Floating point instructions
PAPI_LD_INS	Load instructions
PAPI_SR_INS	Store instructions

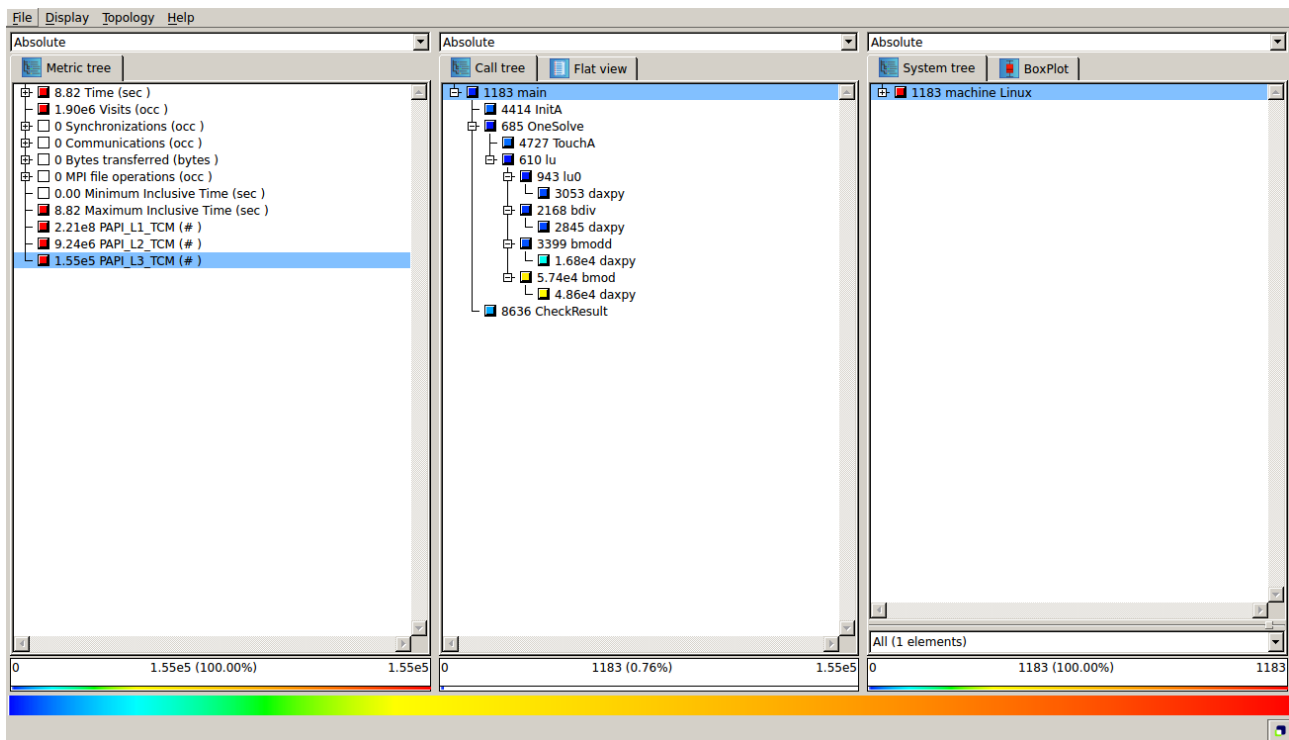
Ο λόγος για τον οποίο επιλέχθηκαν οι παραπάνω είναι ότι τα cache misses , lookaside buffer misses και οι εκτέλεση εντολών load και store είναι αυτά που κυρίως επιβαρύνουν την εκτέση του προγράμματος.

Παρακάτω απεικονίζονται αναλυτικά για κάθε συνάρτηση οι τιμές των μετρητών, επιβεβαιώνοντας ότι τα περισσότερα cache misses TLB misses προκαλούνται στη συνάρτηση daxpy και στη bmod.



TLB misses





cache misses Level 3

Ερώτημα 2ο:

α) Καταρχάς, πρέπει να σημειωθεί ότι δεν είναι δυνατόν να προστεθεί παντού παραλληλοποίηση. Ειδικότερα, υπάρχει εξάρτηση των διαγώνιων μπλοκ, δηλαδή η παραγοντοποίηση LU του πρώτου διαγώνιου μπλοκ πρέπει να προηγηθεί της παραγοντοποίησης του δεύτερου διαγώνιου μπλοκ. Το ίδιο ισχύει και για κάθε επανάληψη του εξωτερικού for της συνάρτησης lu. Συνεπώς, δεν μπορεί να εκτελεστεί παράλληλα ούτε η εξωτερική for της συνάρτησης lu, ούτε και η συνάρτηση lu0 για όλα τα διαγώνια μπλοκ. Εκτός αυτού, είναι αξιοσημείωτο το γεγονός ότι αν και η συνάρτηση daxpy μπορεί να εκτελεστεί παράλληλα, αυτό δεν είναι καθόλου αποδοτικό. Αυτό συμβαίνει γιατί η συνάρτηση daxpy, η οποία εκτελεί πράξεις μεταξύ δύο γραμμών ενός μπλοκ, καλείται αρκετές φορές από την lu0, την bdiv, bmod και την bmodd, με αποτέλεσμα τη σπατάλη αρκετού χρόνου για δημιουργία και καταστροφή νημάτων στη περίπτωση προσθήκης οδηγιών `#pragma omp parallel`. Επιπλέον, αν και ο υπόλοιπος κώδικας μπορεί να παραλληλοποιηθεί, αφού δεν υπάρχουν εξαρτήσεις δεδομένων, η παραλληλοποίηση πολλών επιπέδων μπορεί να φέρει τα αντίθετα αποτελέσματα από τα αναμενόμενα. Επίσης, παρατηρήθηκε ότι η προσθήκη οδηγιών `Omp parallel` στις συναρτήσεις InitA, TouchA, PrintA, CheckResult δεν βελτιώνει καθόλου την απόδοση της εφαρμογής, πράγμα αναμενόμενο αφού η εκτέλεση αυτών των συναρτήσεων επιβαρύνει ελάχιστα το χρόνο εκτέλεσης. Για το λόγο αυτό, επιλέξαμε να προσθέσουμε παραλληλοποίηση μόνο μέσα στη συνάρτηση lu, η οποία είναι η βασικότερη συνάρτηση του προγράμματος. Συγκεκριμένα, στο παραδωτέο κώδικα τόσο η εύρεση των παραγόντων U για τα μητρώα της γραμμής i όσο και η εύρεση των παραγόντων L για τα μητρώα της στήλης i, στην i-οστή επανάληψη του εξωτερικού for, γίνεται παράλληλα. Το ίδιο ισχύει και για την ανανέωση των εσωτερικών μπλοκ, η οποία

γίνεται από τη τελευταία διπλή for της συνάρτησης lu. Σε όλες τις δομές επανάληψης for τις οποίες παραλληλοποιήσαμε επιλέξαμε στατική ανάθεση των επαναλήψεων στα νήματα (`scedule(static)`), καθώς παρατηρήσαμε σχετική μείωση του χρόνου σε σχέση με τη περίπτωση δυναμικής δρομολόγησης. Επιπλέον, αφαιρέσαμε το φράγμα από τη πρώτη εσωτερική for, ώστε όποιο νήμα τελειώσει με τον υπολογισμό των παραγόντων U, να προχωρήσει στον υπολογισμό των παραγόντων L στη δεύτερη εσωτερική for. Με την ίδια λογική, θα μπορούσαμε να ορίσουμε `nowait` και στη δεύτερη οδηγία `#pragma omp for`. Αυτό όμως οδηγεί σε λανθασμένο αποτέλεσμα, καθώς η ανανέωση των εσωτερικών μπλοκ απαιτεί τα στοιχεία των παραγόντων L τα οποία υπολογίζονται στη δεύτερη εσωτερική for. Τέλος, ορίσαμε τις μεταβλητές `il`, `jl private`, ώστε κάθε νήμα να τροποποιεί το δικό του αντίγραφο και τις μεταβλητές `I`, `J` ως `lastprivate` ώστε κάθε νήμα να έχει το δικό του αντίγραφο αλλά στο τέλος να ανανεώνονται και οι κοινές μεταβλητές `I`, `J`.

Με βάση τον αριθμό των `cache misses`, αριθμό των `tlb misses` και χρόνο εκτέλεσης, βρέθηκε ότι το μέγεθος μπλοκ που ελαχιστοποιεί αυτές τις παραμετρους είναι το 200.

β) Έπειτα από τη προσθήκη `simd` εντολών στη συνάρτηση `daxpy` προβλέπεται περαιτέρω μείωση του χρόνου εκτέλεσης του προγράμματος. Αυτό είναι λογικό διότι η γραμμοπράξη γίνεται σε μισό αριθμό επαναλήψεων από ότι χωρίς τη χρήση `simd` εντολών. Ο υποδιπλασιασμός των επαναλήψεων οφείλεται στο ότι σε κάθε καταχωρητή `simd` των 128 bit χωράνε δύο `double` αριθμοί. Πρέπει να σημειωθεί ότι μόνο στη `daxpy` μπορούν να γίνουν οι πράξεις διανυσματικά, καθώς μόνο εκεί γίνονται υπολογισμοί (οι συναρτήσεις `lu0`, `bdiv`, `bmod`, `dmodd` καλούν τη `daxpy` για την εκτέλεση γραμμοπράξεων) και οι επαναλήψεις της `for` είναι ανεξάρτητες μεταξύ τους.

```
root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
  1000 by 1000 Matrix
  200 by 200 Element Blocks

Total execution time: 1.683344 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
  2000 by 2000 Matrix
  200 by 200 Element Blocks

Total execution time: 13.485858 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
  4000 by 4000 Matrix
  200 by 200 Element Blocks

Total execution time: 108.579034 seconds

TESTING RESULTS
TEST PASSED
```

Μετρήσεις σειριακού προγράμματος με -O0 για παραμέτρους :

- -n1000
- -n2000
- -n4000

- ```
root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 871.404966 seconds

TESTING RESULTS
TEST PASSED
```

- -n1000
- -n2000
- -n4000

```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.299244 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 2.495636 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 20.446253 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 167.844155 seconds

TESTING RESULTS
TEST PASSED

```

Μετρήσεις παραλληλοποιημένου (openmp) προγράμματος με -O0 και :

1. με 1 νήμα για παραμέτρους:

- -n1000
- -n2000

```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.688193 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 13.639768 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 109.285554 seconds

TESTING RESULTS
TEST PASSED

```

- -n4000



- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

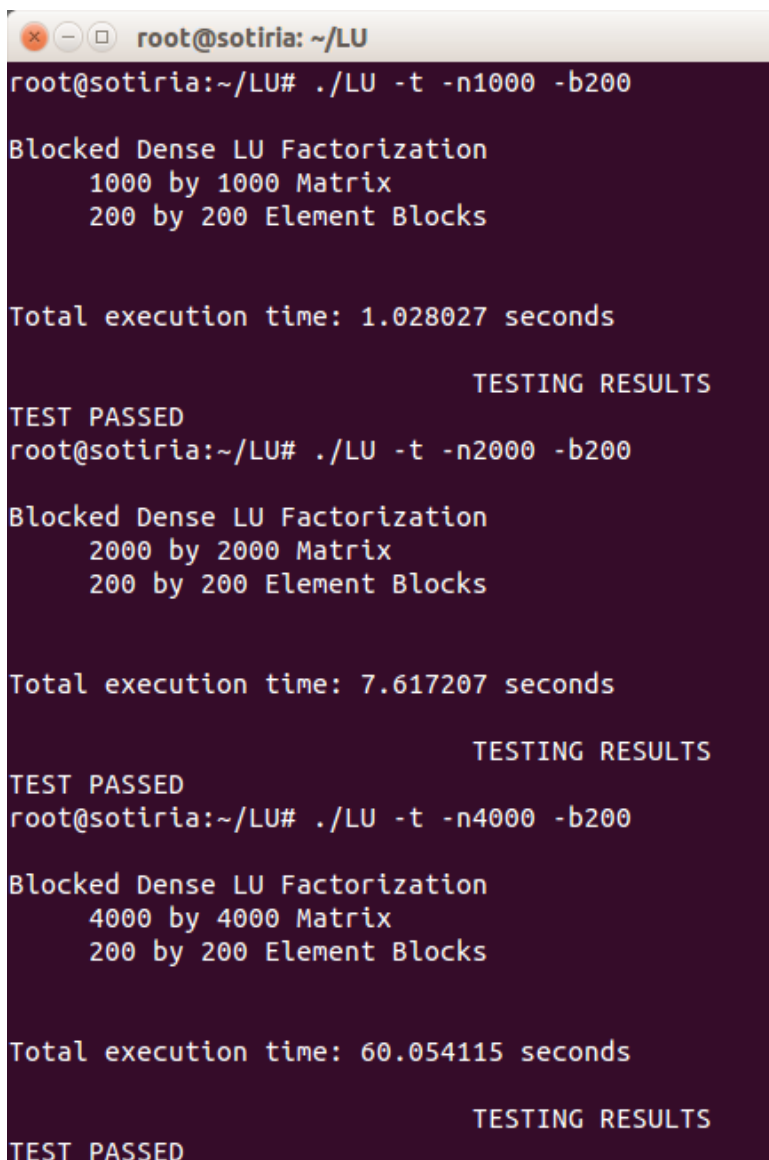
Total execution time: 871.404647 seconds

TESTING RESULTS
TEST PASSED

```

2. με 2 νήματα για παραμέτρους:

- -n1000
- -n2000
- -n4000



```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.028027 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 7.617207 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 60.054115 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 529.630767 seconds

TESTING RESULTS
TEST PASSED

```

3. με 4 νήματα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```

root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.020811 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 7.319917 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 56.460379 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```
root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 467.274426 seconds

TESTING RESULTS
TEST PASSED
```

Μετρήσεις παραλληλοποιημένου (openmp) προγράμματος με -O3 και :

1. με 1 νήμα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```
root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.294236 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 2.450483 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 19.964074 seconds

TESTING RESULTS
TEST PASSED
```

- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 165.090251 seconds

TESTING RESULTS
TEST PASSED

```

2. με 2 νήματα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.198201 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.516917 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 12.244283 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```
root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 103.221745 seconds

TESTING RESULTS
TEST PASSED
```

3. με 4 νήματα για παραμέτρους:

- $-n1000$
- $-n2000$
- $-n4000$

```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.207352 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.479643 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 11.706883 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```
root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 95.136340 seconds

TESTING RESULTS
TEST PASSED
```

Μετρήσεις παραλληλοποιημένου (openmp με simd) προγράμματος με -O3 και :

1. με 1 νήμα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```
root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.408729 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 3.384484 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 27.868828 seconds

TESTING RESULTS
TEST PASSED
```

- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 223.652147 seconds

TESTING RESULTS
TEST PASSED

```

2. με 2 νήματα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```

root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.265869 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.961939 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 15.098657 seconds

TESTING RESULTS
TEST PASSED

```

- -n8000

```

root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 123.369870 seconds

TESTING RESULTS
TEST PASSED

```

3. με 4 νήματα για παραμέτρους:

- -n1000
- -n2000
- -n4000

```

root@sotiria: ~/LU
root@sotiria:~/LU# ./LU -t -n1000 -b200

Blocked Dense LU Factorization
 1000 by 1000 Matrix
 200 by 200 Element Blocks

Total execution time: 0.273397 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n2000 -b200

Blocked Dense LU Factorization
 2000 by 2000 Matrix
 200 by 200 Element Blocks

Total execution time: 1.894576 seconds

TESTING RESULTS
TEST PASSED
root@sotiria:~/LU# ./LU -t -n4000 -b200

Blocked Dense LU Factorization
 4000 by 4000 Matrix
 200 by 200 Element Blocks

Total execution time: 14.846611 seconds

TESTING RESULTS
TEST PASSED

```



- -n8000

```
root@sotiria:~/LU# ./LU -t -n8000 -b200

Blocked Dense LU Factorization
 8000 by 8000 Matrix
 200 by 200 Element Blocks

Total execution time: 122.689272 seconds

TEST PASSED
```

Συμπερασματικά, παρατηρείται σημαντική βελτίωση της απόδοσης της εφαρμογής έπειτα από τη παραλληλοποίηση. Ειδικότερα, το πηλίκο του χρόνου εκτέλεσης του παράλληλου προγράμματος προς το χρόνο εκτέλεσης του σειριακού αυξάνεται με μεγάλο ρυθμό καθώς αυξάνεται το μέγεθος του μητρώου και το πλήθος των νημάτων, φτάνοντας το 50% για μητρώο 8000x8000 και για χρήση 4 νημάτων! Αντιθέτως, με τη προσθήκη SIMD εντολών η απόδοση της παραλληλοποιημένης εφαρμογής μειώθηκε, κάτι που δεν ήταν αναμενόμενο. Βέβαια, η απόδοση της εφαρμογής με χρήση Openmp και Simd εντολών εξακολουθεί να είναι καλύτερη από την απόδοση της σειριακής εφαρμογής.