Παράλληλη Επεξεργασία
Project 2017-2018

Αναστάσιος Χανδρινός ΑΜ:1047171

Δημήτριος Χαραλαμπόπουλος ΑΜ:1047138

Δημήτριος Τάγκαλος ΑΜ: 1047077

Περιεχόμενα

1)Βελτιστοποίηση ακολουθιακού προγραμμάτος

a) Ακολουθιακό πρόγραμμα που δίνεται

O0

--n 1000 –r 350

```
Time for calculations =    712.505404 sec
Time for I/O          =      0.020207 sec
Total execution time  =    712.525611 sec
```

--n 2000 –r 700

```
Time for calculations =   3048.585202 sec
Time for I/O          =      0.031990 sec
Total execution time  =   3048.617192 sec
```

--n 3000 –r 1000

```
Time for calculations =   7082.691306 sec
Time for I/O          =      0.054228 sec
Total execution time  =   7082.745534 sec
```

--n 4000 –r 1300

```
Time for calculations =  11769.532749 sec
Time for I/O          =      0.059699 sec
Total execution time  =  11769.592448 sec
```

--n 5000 –r 1600

```
Time for calculations =  18272.413554 sec
Time for I/O          =      0.074310 sec
Total execution time  =  18272.487864 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    258.361249 sec
Time for I/O          =      0.019318 sec
Total execution time  =    258.380567 sec
```

--n 2000 –r 700

```
Time is 190000
Time for calculations =   1039.167192 sec
Time for I/O          =      0.036094 sec
Total execution time  =   1039.203286 sec
```

--n 3000 –r 1000

```
Time for calculations =   2178.556850 sec
Time for I/O          =      0.045469 sec
Total execution time  =   2178.602319 sec
```

--n 4000 –r 1300

```
Time for calculations =   4218.883309 sec
Time for I/O          =      0.063114 sec
Total execution time  =   4218.946423 sec
```

--n 5000 –r 1600

```
Time for calculations =   6073.325173 sec
Time for I/O          =      0.074492 sec
Total execution time  =   6073.399665 sec
```

b) Ακολουθιακό πρόγραμμα με βελτιστοποίηση αντιγραφών(Βήμα 1.a.)

Ο u και ο uplus είναι pointers και ο καθένας “δείχνει” σε ένα κομμάτι μνήμης. Ουσιαστικά, αντί να αντιγράφουμε τα στοιχεία από το u στο uplus, κάνουμε αντιμετάθεση pointer και ο u δείχνει τα στοιχεία του uplus, ενώ ταυτόχρονα ο uplus κάνει το αντίθετο.

LINE	CODE
329	double *temp=u;
330	u=uplus;
331	uplus=temp;

O0

--n 1000 –r 350

```
Time for calculations =    709.127071 sec
Time for I/O          =      0.017440 sec
Total execution time  =    709.144511 sec
```

--n 2000 –r 700

```
Time for calculations =   2830.964889 sec
Time for I/O          =      0.031100 sec
Total execution time  =   2830.995989 sec
```

--n 3000 –r 1000

```
Time for calculations =   6391.579973 sec
Time for I/O          =      0.044487 sec
Total execution time  =   6391.624460 sec
```

--n 4000 –r 1300

```
Time for calculations =  11364.601889 sec
Time for I/O          =      0.057684 sec
Total execution time  =  11364.659573 sec
```

--n 5000 –r 1600

```
Time for calculations =  17759.082068 sec
Time for I/O          =      0.071193 sec
Total execution time  =  17759.153261 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    246.075469 sec
Time for I/O          =      0.019118 sec
Total execution time  =    246.094587 sec
```

--n 2000 –r 700

```
Time for calculations =    977.032267 sec
Time for I/O          =      0.034830 sec
Total execution time  =    977.067097 sec
```

--n 3000 –r 1000

```
Time for calculations =   2040.006114 sec
Time for I/O          =      0.045230 sec
Total execution time  =   2040.051344 sec
```

--n 4000 –r 1300

```
Time for calculations =   3628.800012 sec
Time for I/O          =      0.058292 sec
Total execution time  =   3628.858304 sec
```

--n 5000 –r 1600

```
Time for calculations =   5660.070288 sec
Time for I/O          =      0.071258 sec
Total execution time  =   5660.141546 sec
```

c) Ακολουθιακό πρόγραμμα με βελτιστοποίηση αντιγραφών + αναδιοργάνωση υπολογισμών (Βήμα 1.a + Βήμα 1.b)

Για την αναδιοργάνωση των υπολογισμών, αναπτύξαμε τον τύπο του αθροίσματος και καταλήξαμε στην παρακάτω λύση, χωρίζοντας το άθροισμα σε δύο κομμάτια. Καταφέραμε να εξοικονομήσουμε χρόνο, παρατηρώντας ότι το Σσij * ui(t) μπορεί να βγει εκτός του triple-nested βρόγχου με τις κατάλληλες απλοποιήσεις.

LINE	CODE
52	double *u, *uplus, *sigma, *omega, *omega1, *new_sigma;

262-266	new_sigma = (double *)calloc(n, sizeof(double));
	if (new_sigma == NULL) {
	printf("Could not allocate memory for \"new_sigma\".\n");
	exit(1);
	}

313-318	for (i=0;i<n;i++){
	new_sigma[i] = 0.0;
	for (j=0;j<n;j++){
	new_sigma[i] += sigma[i*n + j];
	}
	}

335	sum += sigma[i*n + j] * u[j];
337	sum -= new_sigma[i]*u[i];

<u>O0</u>

--n 1000 –r 350

```
Time for calculations =    700.593961 sec
Time for I/O          =      0.018214 sec
Total execution time  =    700.612175 sec
```

--n 2000 –r 700

```
Time for calculations =   2731.389554 sec
Time for I/O          =      0.031117 sec
Total execution time  =   2731.420671 sec
```

--n 3000 –r 1000

```
Time for calculations =   6146.782414 sec
Time for I/O          =      0.044328 sec
Total execution time  =   6146.826742 sec
```

--n 4000 –r 1300

```
Time for calculations =  11035.655160 sec
Time for I/O          =      0.058647 sec
Total execution time  =  11035.713807 sec
```

--n 5000 –r 1600

```
Time for calculations =  16986.358221 sec
Time for I/O          =      0.071716 sec
Total execution time  =  16986.429937 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    244.148787 sec
Time for I/O          =      0.017848 sec
Total execution time  =    244.166635 sec
```

--n 2000 –r 700

```
Time for calculations =    974.552674 sec
Time for I/O          =      0.031658 sec
Total execution time  =    974.584332 sec
```

--n 3000 –r 1000

```
Time for calculations =   2040.251851 sec
Time for I/O          =      0.044728 sec
Total execution time  =   2040.296579 sec
```

--n 4000 –r 1300

```
Time for calculations =   3617.773956 sec
Time for I/O          =      0.058577 sec
Total execution time  =   3617.832533 sec
```

--n 5000 –r 1600

```
Time for calculations =   5654.972150 sec
Time for I/O          =      0.071475 sec
Total execution time  =   5655.043625 sec
```

d. Ακολουθιακό πρόγραμμα με βελτιστοποίηση αντιγραφών + αναδιοργάνωση υπολογισμών + χρήση βιβλιοθήκης BLAS (Βήμα 1.a + Βήμα 1.b + Βήμα 1.c).

Χρησιμοποιήσαμε τη συνάρτηση cblas_dgemv (πολλαπλασιασμός μητρώου x διάνυσμα), πάνω στα μητρώα που δημιουργήθηκαν με τις βελτιστοποιήσεις του προηγούμενου ερωτήματος.

LINE	CODE
7	#include <cblas.h>
334	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
335	for (i = 0; i < n; i++) {
336	uplus[i] = u[i] + dt * (mu - u[i]);
*	δεν υπάρχει πια ο κόμβος j

<u>O0</u>

--n 1000 –r 350

```
Time for calculations =    239.632018 sec
Time for I/O          =      0.023146 sec
Total execution time  =    239.655164 sec
```

--n 2000 –r 700

```
Time for calculations =   1184.185719 sec
Time for I/O          =      0.041920 sec
Total execution time  =   1184.227639 sec
```

--n 3000 –r 1000

```
Time for calculations =   1937.328232 sec
Time for I/O          =      0.051789 sec
Total execution time  =   1937.380021 sec
```

--n 4000 –r 1300

```
Time for calculations =   3342.914874 sec
Time for I/O          =      0.066690 sec
Total execution time  =   3342.981564 sec
```

--n 5000 –r 1600

```
Time for calculations =   5198.723457 sec
Time for I/O          =      0.081049 sec
Total execution time  =   5198.804506 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    165.972975 sec
Time for I/O          =      0.019874 sec
Total execution time  =    165.992849 sec
```

--n 2000 –r 700

```
Time for calculations =    870.067226 sec
Time for I/O          =      0.037573 sec
Total execution time  =    870.104799 sec
```

--n 3000 –r 1000

```
Time for calculations =   1882.272192 sec
Time for I/O          =      0.052573 sec
Total execution time  =   1882.324765 sec
```

--n 4000 –r 1300

```
Time for calculations =   3230.656696 sec
Time for I/O          =      0.066983 sec
Total execution time  =   3230.723679 sec
```

--n 5000 –r 1600

```
Time for calculations =   5237.711857 sec
Time for I/O          =      0.082635 sec
Total execution time  =   5237.794492 sec
```

f. Δώστε ιδιαίτερη βάση στις μετρήσεις με και χωρίς βελτιστοποιήσεις του μεταφραστή. Τι επίπτωση έχει κάθε βελτιστοποίηση που κάνετε εσείς (Βήματα 1.a, 1.b, 1.c) στον χρόνο εκτέλεσης του προγράμματος όταν χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή και όταν δεν χρησιμοποιούνται;

Ουσιαστικά, όταν χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή(O3), ενεργοποιείται η αυτόματη διανυσματοποίηση, η οποία οδηγεί σε ταχύτερους χρόνους εκτέλεσης του προγράμματος.

- Παρατήρουμε, ότι οι βελτιστοποιήσεις μας στο 1a, όταν δε χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή, είναι κατά μέσο όρο 4.1%[1].
- Όταν, βέβαια, χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή έχουμε ποσοστιαία βελτιστοποίηση της τάξεως του 6.6%.
- Όσο αφορά το 1b, βλέπουμε, ότι χωρίς τις βελτιστοποιήσεις του μεταφραστή, υπάρχει βελτιστοποίηση 3.2%, σε σχέση με αυτές του 1a.
- Αντίθετα, όταν τρέχουν όλες οι βελτιστοποιήσεις του μεταφραστή στο 1b, έχουμε βελτίωση κατά 1.3%.
- Τέλος, για το 1c, δίχως τις βελτιστοποιήσεις του μεταφραστή, υπάρχει καλυτέρευση σε σχέση με το 1b, κατά 63.4%.
- Με τις βελτιστοποιήσεις του μεταφραστή όμως, έχουμε βελτίωση μόνον 13,2%.

[1]

Όλες οι μετρήσεις έγιναν κατά προσέγγιση πρώτου δεκαδικού ψηφίου.

2) Παραλληλοποιημένος κώδικας (βασιζόμενοι στο ακολουθιακό πρόγραμμα που προέκυψε από το Βήμα 1.b)

a. Παραλληλοποίηση μόνο του βρόχου “it”

Για την παραλληλοποίηση χρησιμοποιήσαμε την οδηγία #pragma omp parallel private(it,j, sum). Με αυτόν τον τρόπο οι μεταβλητές it,j και sum είναι private, που σημαίνει ότι κάθε thread έχει το δικό του αντίγραφο της εκάστοτε μεταβλητής. Στο τέλος του loop, εφαρμόσαμε #pragma omp single, υποδηλώνοντας ότι το παρών block εκτελείται μόνο από 1 thread (χωρίς να διευκρινίζεται ποιο). Τα υπόλοιπα threads απλά προσπερνούν το παρών block περιμένοντας κάποιο "εμπόδιο" στο τέλος του block.

LINE	CODE
324	#pragma omp parallel private(it,j, sum)
325	{
330	#pragma omp for
349	#pragma omp single
350	{
394	}
395	}

1 thread

O0

--n 1000 –r 350

```
Time for calculations =    754.133184 sec
Time for I/O          =      0.018917 sec
Total execution time  =    754.152101 sec
```

--n 2000 –r 700

```
Time for calculations =   2993.112018 sec
Time for I/O          =      0.033726 sec
Total execution time  =   2993.145744 sec
```

--n 3000 –r 1000

```
Time for calculations =   6765.783283 sec
Time for I/O          =      0.046736 sec
Total execution time  =   6765.830019 sec
```

--n 4000 –r 1300

```
Time for calculations =  11825.516422 sec
Time for I/O          =      0.058760 sec
Total execution time  =  11825.575182 sec
```

--n 5000 –r 1600

```
Time for calculations =  34945.182198 sec
Time for I/O          =      0.072743 sec
Total execution time  =  34945.254941 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    237.609643 sec
Time for I/O          =      0.018054 sec
Total execution time  =    237.627697 sec
```

--n 2000 –r 700

```
Time for calculations =   1172.443074 sec
Time for I/O          =      0.038787 sec
Total execution time  =   1172.481861 sec
```

--n 3000 –r 1000

```
Time for calculations =   2039.669361 sec
Time for I/O          =      0.046146 sec
Total execution time  =   2039.715507 sec
```

--n 4000 –r 1300

```
Time for calculations =   3614.807049 sec
Time for I/O          =      0.059900 sec
Total execution time  =   3614.866949 sec
```

--n 5000 –r 1600

```
Time for calculations =   5916.327856 sec
Time for I/O          =      0.075425 sec
Total execution time  =   5916.403281 sec
```

2 threads

O0

--n 1000 –r 350

```
ime for calculations =    477.469111 sec
ime for I/O          =      0.024927 sec
otal execution time  =    477.494038 sec
```

--n 2000 –r 700

```
Time for calculations =   1866.756372 sec
Time for I/O          =      0.034748 sec
Total execution time  =   1866.791120 sec
```

--n 3000 –r 1000

```
Time for calculations =   4108.646118 sec
Time for I/O          =      0.049336 sec
Total execution time  =   4108.695454 sec
```

--n 4000 –r 1300

```
Time for calculations =   7226.275111 sec
Time for I/O          =      0.065164 sec
Total execution time  =   7226.340275 sec
```

--n 5000 –r 1600

```
Time for calculations =  11187.403174 sec
Time for I/O          =      0.077370 sec
Total execution time  =  11187.480544 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    129.296638 sec
Time for I/O          =      0.018193 sec
Total execution time  =    129.314831 sec
```

--n 2000 –r 700

```
Time for calculations =    579.894531 sec
Time for I/O          =      0.037830 sec
Total execution time  =    579.932361 sec
```

--n 3000 –r 1000

```
Time for calculations =   1258.043852 sec
Time for I/O          =      0.054353 sec
Total execution time  =   1258.098205 sec
```

--n 4000 –r 1300

```
Time for calculations =   2230.549243 sec
Time for I/O          =      0.072148 sec
Total execution time  =   2230.621391 sec
```

--n 5000 –r 1600

```
Time for calculations =   3688.401387 sec
Time for I/O          =      0.093563 sec
Total execution time  =   3688.494950 sec
```

4 threads

O0

--n 1000 –r 350

```
Time for calculations =    480.267656 sec
Time for I/O          =      0.039683 sec
Total execution time  =    480.307339 sec
```

--n 2000 –r 700

```
Time for calculations =   2192.917693 sec
Time for I/O          =      0.062115 sec
Total execution time  =   2192.979808 sec
```

--n 3000 –r 1000

```
Time for calculations =   4693.017417 sec
Time for I/O          =      0.079201 sec
Total execution time  =   4693.096618 sec
```

--n 4000 –r 1300

```
Time for calculations =   7627.314104 sec
Time for I/O          =      0.095224 sec
Total execution time  =   7627.409328 sec
```

--n 5000 –r 1600

```
Time for calculations =  12197.033207 sec
Time for I/O          =      0.115663 sec
Total execution time  =  12197.148870 sec
```

O3

--n 1000 –r 350

```
Time for calculations =     86.823048 sec
Time for I/O          =      0.035186 sec
Total execution time  =     86.858234 sec
```

--n 2000 –r 700

```
Time for calculations =    422.962162 sec
Time for I/O          =      0.060052 sec
Total execution time  =    423.022214 sec
```

--n 3000 –r 1000

```
Time for calculations =    809.708326 sec
Time for I/O          =      0.080285 sec
Total execution time  =    809.788611 sec
```

--n 4000 –r 1300

```
Time for calculations =   1501.782700 sec
Time for I/O          =      0.104566 sec
Total execution time  =   1501.887266 sec
```

--n 5000 –r 1600

```
Time for calculations =   2980.118970 sec
Time for I/O          =      0.137423 sec
Total execution time  =   2980.256393 sec
```

b. Παραλληλοποίηση μόνο του βρόχου “i”

Για την παραλληλοποίηση αυτού του loop, εφαρμόσθηκε η εντολή #pragma omp parallel for private(i,sum,j).Με αυτό τον τρόπο οι μεταβλητές i,sum και j είναι private, που σημαίνει ότι κάθε thread έχει το δικό του αντίγραφο της εκάστοτε μεταβλητής.

LINE	CODE
328	#pragma omp parallel for private(i,sum,j)

1 thread

O0

--n 1000 –r 350

```
Time for calculations =    745.549378 sec
Time for I/O          =      0.021495 sec
Total execution time  =    745.570873 sec
```

--n 2000 –r 700

```
Time for calculations =   2841.749729 sec
Time for I/O          =      0.032657 sec
Total execution time  =   2841.782386 sec
```

--n 3000 –r 1000

```
Time for calculations =   6399.463342 sec
Time for I/O          =      0.046357 sec
Total execution time  =   6399.509699 sec
```

--n 4000 –r 1300

```
Time for calculations =  11363.132602 sec
Time for I/O          =      0.059555 sec
Total execution time  =  11363.192157 sec
```

--n 5000 –r 1600

```
Time for calculations =  17683.793184 sec
Time for I/O          =      0.071548 sec
Total execution time  =  17683.864732 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    233.485257 sec
Time for I/O          =      0.021841 sec
Total execution time  =    233.507098 sec
```

--n 2000 –r 700

```
Time for calculations =    898.061098 sec
Time for I/O          =      0.034813 sec
Total execution time  =    898.095911 sec
```

--n 3000 –r 1000

```
Time for calculations =   1985.351358 sec
Time for I/O          =      0.046848 sec
Total execution time  =   1985.398206 sec
```

--n 4000 –r 1300

```
Time for calculations =   3500.037601 sec
Time for I/O          =      0.059385 sec
Total execution time  =   3500.096986 sec
```

--n 5000 –r 1600

```
Time for calculations =   5628.867297 sec
Time for I/O          =      0.074769 sec
Total execution time  =   5628.942066 sec
```

2 threads

O0

--n 1000 –r 350

```
Time for calculations =    477.046720 sec
Time for I/O          =      0.026391 sec
Total execution time  =    477.073111 sec
```

--n 2000 –r 700

```
Time for calculations =   1776.707698 sec
Time for I/O          =      0.035797 sec
Total execution time  =   1776.743495 sec
```

--n 3000 –r 1000

```
Time for calculations =   4005.612765 sec
Time for I/O          =      0.054103 sec
Total execution time  =   4005.666868 sec
```

--n 4000 –r 1300

```
Time for calculations =   7105.948326 sec
Time for I/O          =      0.067754 sec
Total execution time  =   7106.016080 sec
```

--n 5000 –r 1600

```
Time for calculations =  12235.529959 sec
Time for I/O          =      0.092647 sec
Total execution time  =  12235.622606 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    133.925141 sec
Time for I/O          =      0.018318 sec
Total execution time  =    133.943459 sec
```

--n 2000 –r 700

```
Time for calculations =    577.412096 sec
Time for I/O          =      0.036746 sec
Total execution time  =    577.448842 sec
```

--n 3000 –r 1000

```
Time for calculations =   1255.743595 sec
Time for I/O          =      0.053996 sec
Total execution time  =   1255.797591 sec
```

--n 4000 –r 1300

```
Time for calculations =   2217.636277 sec
Time for I/O          =      0.071017 sec
Total execution time  =   2217.707294 sec
```

--n 5000 –r 1600

```
Time for calculations =   3646.519166 sec
Time for I/O          =      0.090912 sec
Total execution time  =   3646.610078 sec
```

4 threads

O0

--n 1000 –r 350

```
Time for calculations =    473.810474 sec
Time for I/O          =      0.038241 sec
Total execution time  =    473.848715 sec
```

--n 2000 –r 700

```
Time for calculations =   1887.182663 sec
Time for I/O          =      0.058758 sec
Total execution time  =   1887.241421 sec
```

--n 3000 –r 1000

```
Time for calculations =   4262.193399 sec
Time for I/O          =      0.075281 sec
Total execution time  =   4262.268680 sec
```

--n 4000 –r 1300

```
Time for calculations =   8415.671323 sec
Time for I/O          =      0.098653 sec
Total execution time  =   8415.769976 sec
```

--n 5000 –r 1600

```
Time for calculations =  12000.695109 sec
Time for I/O          =      0.111219 sec
Total execution time  =  12000.806328 sec
```

O3

--n 1000 –r 350

```
Time for calculations =     92.481943 sec
Time for I/O          =      0.039136 sec
Total execution time  =     92.521079 sec
```

--n 2000 –r 700

```
Time for calculations =    402.803122 sec
Time for I/O          =      0.062123 sec
Total execution time  =    402.865245 sec
```

--n 3000 –r 1000

```
Time for calculations =    765.829751 sec
Time for I/O          =      0.078493 sec
Total execution time  =    765.908244 sec
```

--n 4000 –r 1300

```
Time for calculations =   1384.297611 sec
Time for I/O          =      0.092668 sec
Total execution time  =   1384.390279 sec
```

--n 5000 –r 1600

```
Time for calculations =   2222.868472 sec
Time for I/O          =      0.111704 sec
Total execution time  =   2222.980176 sec
```

c. Παραλληλοποίηση μόνο του βρόχου “j”

Στο συγκεκριμένο loop, παρατηρούμε ότι υπάρχει εξάρτηση δεδομένων, καθώς ο υπολογισμός σε μια επανάληψη του βρόγχου εξαρτάται από αποτελέσματα προηγουμένων επαναλήψεων. Σε αυτή την περίπτωση, χρησιμοποιήθηκε η οδηγία #pragma omp parallel for reduction(+:sum). Αυτο που επιτυγχάνει η προαναφερθείσα εντολή είναι να "διατάζει" τον compiler να χρησιμοποιεί τιμές από διαφορετικές επαναλήψεις του βρόγχου(στην περίπτωση μας, τιμές του sum από προηγούμενες επαναλήψεις).

LINE	CODE
335	#pragma omp parallel for reduction(+:sum)

1 thread

O0

--n 1000 –r 350

```
Time for calculations =    824.854297 sec
Time for I/O          =      0.019122 sec
Total execution time  =    824.873419 sec
```

--n 2000 –r 700

```
Time for calculations =   3093.480823 sec
Time for I/O          =      0.031593 sec
Total execution time  =   3093.512416 sec
```

--n 3000 –r 1000

```
Time for calculations =   6769.684627 sec
Time for I/O          =      0.044225 sec
Total execution time  =   6769.728852 sec
```

--n 4000 –r 1300

```
Time for calculations =  11905.794513 sec
Time for I/O          =      0.060810 sec
Total execution time  =  11905.855323 sec
```

--n 5000 –r 1600

```
Time for calculations =  18379.121915 sec
Time for I/O          =      0.071606 sec
Total execution time  =  18379.193521 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    357.759097 sec
Time for I/O          =      0.019911 sec
Total execution time  =    357.779008 sec
```

--n 2000 –r 700

```
Time for calculations =   1175.616975 sec
Time for I/O          =      0.033094 sec
Total execution time  =   1175.650069 sec
```

--n 3000 –r 1000

```
Time for calculations =   2455.345641 sec
Time for I/O          =      0.044947 sec
Total execution time  =   2455.390588 sec
```

--n 4000 –r 1300

```
Time for calculations =   4176.764606 sec
Time for I/O          =      0.058559 sec
Total execution time  =   4176.823165 sec
```

--n 5000 –r 1600

```
Time for calculations =   6246.140354 sec
Time for I/O          =      0.072177 sec
Total execution time  =   6246.212531 sec
```

2 threads

O0

--n 1000 –r 350

```
Time for calculations =    727.054918 sec
Time for I/O          =      0.026295 sec
Total execution time  =    727.081213 sec
```

--n 2000 –r 700

```
Time for calculations =   2385.253447 sec
Time for I/O          =      0.036209 sec
Total execution time  =   2385.289656 sec
```

--n 3000 –r 1000

```
Time for calculations =   4967.540934 sec
Time for I/O          =      0.052771 sec
Total execution time  =   4967.593705 sec
```

--n 4000 –r 1300

```
Time for calculations =   8267.048992 sec
Time for I/O          =      0.067034 sec
Total execution time  =   8267.116026 sec
```

--n 5000 –r 1600

```
Time for calculations =  12391.750430 sec
Time for I/O          =      0.076410 sec
Total execution time  =  12391.826840 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    412.238841 sec
Time for I/O          =      0.022922 sec
Total execution time  =    412.261763 sec
```

--n 2000 –r 700

```
Time for calculations =   1123.704917 sec
Time for I/O          =      0.039653 sec
Total execution time  =   1123.744570 sec
```

--n 3000 –r 1000

```
Time for calculations =   2099.649939 sec
Time for I/O          =      0.052273 sec
Total execution time  =   2099.702212 sec
```

--n 4000 –r 1300

```
Time for calculations =   3408.176863 sec
Time for I/O          =      0.069524 sec
Total execution time  =   3408.246387 sec
```

--n 5000 –r 1600

```
Time for calculations =   4674.898596 sec
Time for I/O          =      0.082405 sec
Total execution time  =   4674.981001 sec
```

4 threads

O0

--n 1000 –r 350

```
Time for calculations =   1164.954270 sec
Time for I/O          =      0.035313 sec
Total execution time  =   1164.989583 sec
```

--n 2000 –r 700

```
Time for calculations =   2170.730308 sec
Time for I/O          =      0.053133 sec
Total execution time  =   2170.783441 sec
```

--n 3000 –r 1000

```
Time for calculations =   4542.177400 sec
Time for I/O          =      0.070792 sec
Total execution time  =   4542.248192 sec
```

--n 4000 –r 1300

```
Time for calculations =   7694.575556 sec
Time for I/O          =      0.084174 sec
Total execution time  =   7694.659730 sec
```

--n 5000 –r 1600

```
Time for calculations =  11670.501930 sec
Time for I/O          =      0.097742 sec
Total execution time  =  11670.599672 sec
```

O3

--n 1000 –r 350

```
Time for calculations =    398.688421 sec
Time for I/O          =      0.034235 sec
Total execution time  =    398.722656 sec
```

--n 2000 –r 700

```
Time for calculations =    980.357683 sec
Time for I/O          =      0.053419 sec
Total execution time  =    980.411102 sec
```

--n 3000 –r 1000

```
Time for calculations =   2270.181668 sec
Time for I/O          =      0.073012 sec
Total execution time  =   2270.254680 sec
```

--n 4000 –r 1300

```
Time for calculations =   2537.667885 sec
Time for I/O          =      0.081781 sec
Total execution time  =   2537.749666 sec
```

--n 5000 –r 1600

```
Time for calculations =   3515.275470 sec
Time for I/O          =      0.099537 sec
Total execution time  =   3515.375007 sec
```

d. Κρατήστε το καλύτερο από τις παραπάνω 3 περιπτώσεις (όταν χρησιμοποιείτε μέγιστες βελτιστοποιήσεις –O3)

i. Προσθέστε χρήση βιβλιοθήκης BLAS

Για το εν λόγω ερώτημα σε πρώτη φάση, προσπαθήσαμε να κάνουμε υλοποίηση για το βρόγχο “i”, γιατί αρχικά δεν είχαμε καταφέρει να παραλληλοποίησουμε σωστά το βρόγχο “it”, και ο “i” έβγαινε ο πιο γρήγορος. Αυτό το κάναμε, βάζοντας τη συνάρτηση cblas_dgemv πριν την παραλληλοποίηση του βρόγχου με #pragma omp parallel for private(i,j). Βέβαια, σωστή παραλληλοποίηση δε γνωρίζουμε αν έγινε και οι χρόνοι δεν ήταν και οι καλύτεροι δυνατοί.

LINE	CODE
334	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
335	#pragma omp parallel for private(i,j)

4 threads O3 –n 200 --r 60

```
Time for calculations =     11.348231 sec
Time for I/O          =      0.011124 sec
Total execution time  =     11.359355 sec
```

Αυτό προσπαθήσαμε να το πετύχουμε με τη χρήση της εντολής

export USE_OPENMP=1

στο shell πριν την εκτέλεση της μέτρησης, αλλά οι διαφορές κάποιες φορές φάνηκαν και κάποιες όχι, άρα δεν είμαστε σίγουροι για την ορθότητα της.

4 threads O3 –n 200 --r 60

```
Time for calculations =     11.192804 sec
Time for I/O          =      0.011506 sec
Total execution time  =     11.204310 sec
```

Σε δεύτερη φάση, όταν τελικά καταφέραμε να παραλληλοποίησουμε τον βρόγχο “it”, και καταλήξαμε ότι αυτός είναι ο ταχύτερος, προσπαθήσαμε να βάλουμε τη cblas_dgemv στο βρόγχο αυτό με τον παρακάτω τρόπο, όμως δυστυχώς καταφέραμε να το κάνουμε να λειτουργεί σωστά(με equal στη σύγκριση nummdif) για ένα και μόνο thread. Η λογική μας ήταν, κάθε διαφορετικό thread, να υπολογίζει ένα διαφορετικό μέρος του πολλαπλασιασμού. Αυτό, όπως φαίνεται παρακάτω, προσπαθήσαμε να το υλοποιήσουμε με τη χρήση μιας switch που δέχεται ως όρισμα τον αριθμό του thread που δουλεύει τη δεδομένη στιγμή (tid). Επί της ουσίας κάθε διαφορετικό case θα διαχειρίζεται διαφορετικό μέρος του πολλαπλασιασμού τρέχοντας τη cblas_dgemv, για διαφορετικά κομμάτια του μητρώου. Δυστυχώς, για λόγους χρονικούς, και όχι μόνο, δεν καταφέραμε να υλοποιήσουμε σωστά αυτή την υπόθεση, να θέσουμε δηλαδή τα κατάλληλα ορίσματα σε κάθε κάλεσμα της συνάρτησης.

LINE	CODE
1	#include <cblas.h>
7	#include <omp.h>
333	#pragma omp parallel private(it,i) shared(sigma, u,sum)
334	{
339	int tid = omp_get_thread_num() ;
340	switch (tid)
341	{
342	case 0: //thread 0
343	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
344	break;
345	case 1: //thread 1
346	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
347	break;
348	case 2: //thread 2
349	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
350	break;
351	case 3: //thread 4
352	cblas_dgemv(CblasColMajor, CblasNoTrans, n, n, 1, sigma, n, u, 1, 0, sum, 1);
353	break;
354	default:
355	break;
356	}
357	#pragma omp for
376	#pragma omp single
377	{
420	}
422	}

1 thread O3 –n 200 --r 60

```
Time for calculations =      3.640668 sec
Time for I/O          =      0.005162 sec
Total execution time  =      3.645830 sec
```

Η ορθότητα των αποτελεσμάτων, σε κάθε περίπτωση μέχρι αυτό το σημείο του project, έχει ελεγχθεί επιτυχώς, μέσω nummdif.

f. Δώστε ιδιαίτερη βάση στις μετρήσεις με και χωρίς βελτιστοποιήσεις του μεταφραστή. Τι επίπτωση έχει κάθε βελτιστοποίηση που κάνετε εσείς (Βήματα 2.a, 2.b, 2.c, 2.d) στον χρόνο εκτέλεσης του προγράμματος όταν χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή και όταν δεν χρησιμοποιούνται;[2]

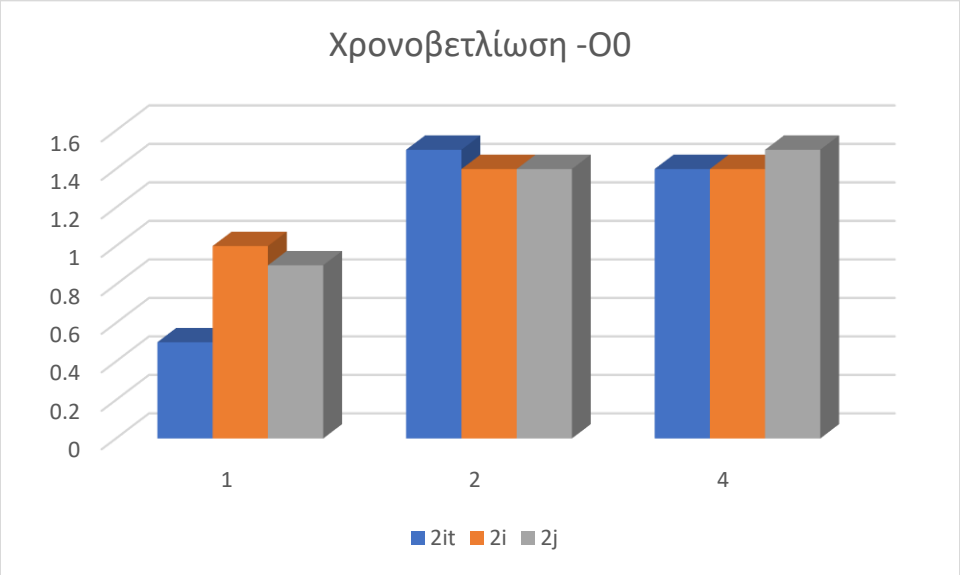
- Παρατηρούμε, ότι η παραλληλοποίηση στο βρόχο it (ερωτημα 2.a), όταν δεν χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή, οδηγεί σε βελτίωση του χρόνου κατά 29,2%. Όταν δε, χρησιμοποιούνται και οι βελτιστοποιήσεις του μεταφραστή, έχουμε βελτίωση κατά 57,7%.
- Όσο αναφορά την παραλληλοποίηση του βρόχου i(ερώτημα 2.b), όταν δεν λαμβάνονται υπόψιν οι βελτιστοποιήσεις του μεταφραστή, παρατηρούμε βελτίωση του χρόνου 31,7% κατά μέσο όρο. Αντίθετα, όταν χρησιμοποιούνται οι βελτιστοποιήσεις του μεταφραστή, υπάρχει ποσοστιαία βελτιστοποίηση της τάξεως του 64,1%.
- Τέλος, στην παραλληλοποίηση του βρόχου j(ερώτημα 2.c), χωρίς τις βελτιστοποιήσεις του μεταφραστή, η βελτίωση στον χρόνο κυμαίνεται γύρω στις 15,8 ποσοστιαίες μονάδες. Όταν χρησιμοποιούνται και οι βελτιστοποιήσεις του μεταφραστή, η ποσοστιαία βελτίωση στον χρόνο είναι μόνο 3,5%. Βλέποντας τις ποσοστιαίες μεταβολές του βρόχου j(τόσο με όσο και χωρίς τις βελτιστοποιήσεις του μεταφραστή), είναι εύκολο κανείς να συμπεράνει ότι η παραλληλοποίηση του βρόχου j, δεν είναι η αποδοτικότερη.[2]
- Δε θεωρήσαμε σκόπιμο να συγκρίνουμε τις μετρήσεις για το 2.d, καθώς η προσπάθεια επίλυσης που κάναμε, δεν έδινε αρκετά στοιχεία.

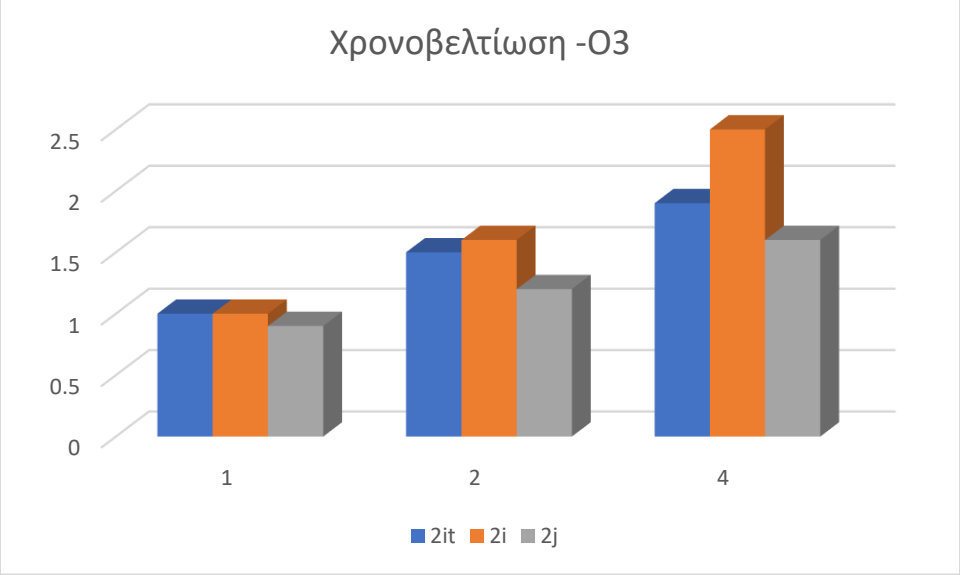
[2] Οι συγκρίσεις έγιναν μεταξύ των αποτελεσμάτων του ερωτήματος 1b, και των αποτελεσμάτων της παραλληλοποίησης του εκάστοτε loop, με την χρήση 4ων threads και κατά προσέγγιση πρώτου δεκαδικού ψηφίου.

Διάγραμματα χρονοβελτίωσης[3]

Για τα διαγράμματα χρησιμοποιήθηκε ο τύπος

speedup = sequential time / parallel time





3) Στο καλύτερο παράλληλο πρόγραμμα που προέκυψε

a. Βελτιστοποίηση εγγραφής αποτελεσμάτων σε αρχείο.

Στο συγκεκριμένο ερώτημα πήραμε μόνο μία μέτρηση (–n 1000 –r350) απ’ το 2b μας, μεταγλωττίζοντας το με την -DALL_RESULTS παράμετρο, καθώς το πρόγραμμα γέμιζε το δίσκο μας σε μεγαλύτερες τιμές, και δεν είχαμε σιγουριά για την ορθότητα των αποτελεσμάτων.

```
Time for calculations =   1167.319445 sec
Time for I/O          =    360.278506 sec
Total execution time  =   1527.597951 sec
```

Τώρα, όσον αφορά την υλοποίηση τεχνικών για την μείωση του χρόνου που απαιτείται για την αποθήκευση αποτελεσμάτων σε ένα παράλληλο πρόγραμμα, για βελτιστοποίηση των I/O χρόνων, αποφασίσαμε να χρησιμοποιήσουμε την fflush(stdout), για το buffering των printf του προγράμματος. Για τη μέτρηση, πράξαμε ομοίως με αυτήν του 2b με -DALL_RESULTS.

LINE	CODE
402	fprintf(output1, "%ld\t", it);
403	fflush(stdout);
405	fprintf(output1, "%19.15f", u[i]);
406	fflush(stdout);
408	fprintf(output1, "\n");
409	fflush(stdout);
412	fprintf(output2, "%ld\t", it);
413	fflush(stdout);
416	fprintf(output2, "%19.15f", omega[i]);
417	fflush(stdout);
419	fprintf(output2, "\n");
420	fflush(stdout);

```
Time for calculations =   1131.172268 sec
Time for I/O          =    354.001621 sec
Total execution time  =   1485.173889 sec
```

Ευχαριστούμε πολύ για την προσοχή σας.