ΠΑΡΑΛΛΗΛΑ

ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ 2018 – 2019

“Συνέλιξη Εικόνων”

SHAVLEGO MNATOBISHVILI ***ΑΜ: 1115201200113***

ΔΗΜΗΤΡΙΟΣ ΚΟΤΣΑΝΙΤΗΣ ***ΑΜ: 1115201400254***

Εισαγωγικά

Στην παρούσα εργασία, κληθήκαμε να υλοποιήσουμε την εφαρμογή της συνέλιξης μιας εικόνας, η οποία και είναι πολύ χρήσιμη σε πολύ επίκαιρα προγράμματα της καθημερινότητάς μας που αφορούν την επεξεργασία φωτογραφίας. Σε ό,τι αφορά το μάθημά μας, η συνάφεια προκύπτει από την μελέτη του συγκεκριμένου προγράμματος σχετικά με την απόδοσή του τρέχοντας σειριακά αλλά και κατανεμημένα σε παράλληλα συστήματα και από την εξαγωγή συμπερασμάτων σχετικά με το ποια είναι η βέλτιστη επιλογή που μπορούμε να ακολουθήσουμε

δεδομένων ορισμένων παραμέτρων. . Το πρόγραμμα αναπτύχθηκε στη γλώσσα C και για την παραλληλία χρησιμοποιήθηκε MPI. Για να εξεταστεί η αποδοτικότητα της παραλληλίας που άλλωστε είναι και το ζητούμενο της εν λόγω άσκησης, έγιναν μετρήσεις στα Linux μηχανήματα της σχολής και υπολογίστηκαν οι χρόνοι εκτέλεσης των προγραμμάτων και συνεπακολούθως και τα Speedup και Efficiency. Η άσκηση υλοποιήθηκε με απόλυτη λειτουργικότητα, τουλάχιστον κατά το δικό μας έλεγχο. Στο παραδοτέο περιέχονται 4 αρχεία .c για υλοποιήσεις MPI και MPI OpenMP με και χωρίς συγχρονισμό, το αρχείο machines.txt που φτίαχτηκε για να τρέχουμε στα μηχανήματα της σχολής τα προγράμματά μας, το entoles.txt που αναγράφονται ενδεικτικά οι εντολές μεταγλώττισης των αρχείων κώδικα καθώς και το παρόν readme.

Πιο συγκεκριμένα

Η άσκηση υλοποιήθηκε με τη δυνατότητα εισόδου φωτογραφίας σαν input. Η φωτογραφία που είσάγουμε, λοιπόν, υπάρχει τόσο σε έγχρωμη μορφή όσο και σε ασπρόμαυρη και είναι αυτή που υπάρχει στο δοσμένο waterfall.rar. Επιλέξαμε σαν διαστάσεις της εικόνας το ένα τέταρτο, το ένα δεύτερο, την original διάστασή της, τη διπλάσια και την τετραπλάσια διάστασή της. Πιο αναλυτικά δηλαδή τα **μεγέθη** είναι :

1.960*1260 (*1/4)

2.1920*1260 (*1/2)

3.1920*2520 (*1)

4.3840*2520 (*2)

5. 3840*5040(*4)

Επίσης, για την λειτουργία του παράλληλου κώδικα το πρόγραμμα μοιράζεται σε 1 (δηλαδή σειριακή λειτουργία), 4, 9, 16, 25 και 36 **διεργασίες**. Σε ό,τι αφορά την εφαρμογή του φίλτρου της συνέλιξης, που συνιστά και μια **γενιά,** οι μετρήσεις τις οποίες λάβαμε αφορούν την εφαρμογή 50 γενιών.

Σχόλια υλοποίησης

1.Τρέχοντας το εκτελέσιμο, γίνεται έλεγχος για τα ορίσματα ότι έχουν δοθεί σωστά και αφού γίνουν οι απαραίτητοι έλεγχοι από τη master process, κάνει broadcast στις υπόλοιπες τις παραμέτρους εκτέλεσης.

2.Ανάλογα με τον αριθμό των διεργασιών γίνεται διαμέριση του φόρτου ώστε η κάθε μια να διαβάζει παράλληλα με τις άλλες το κομμάτι που της αναλογεί. Αντίστοιχα στο τέλος γίνεται και το παράλληλο γράψιμο της καινούριας εικόνας.

3.Με ελέγχους για τον τύπο της εικόνας κάθε φορά εκτελείται άλλο κομμάτι για να γίνεται σωστά η συνέλιξη, ανάλογα με το όρισμα για το τρέξιμο του προγράμματος (grey/rgb).

4.Ο πίνακας έχει δεσμευτεί μονοδιάστατος και όχι δισδιάστατος καθώς η κωδικοποίηση του προβλήματος αποδείχτηκε αποτελεσματικότερη με τη χρήση αυτού του προτύπου. Επιπρόσθετα, η διαμέριση διευκολύνεται σημαντικά με τη χρήση του μονοδιάστατου μοντέλου

5.Σε αυτό το στάδιο προσπαθούμε να εξάγουμε τρόπους παραλληλισμού της εκτέλεσης. Ένας καλός διαμερισμός διαιρεί σε μικρά κομμάτια τόσο τους υπολογισμούς όσο και τα δεδομένα του προβλήματος. Έτσι χωρίζουμε την εικόνα σε πιο μικρές εικόνες ανάλογα με τον αριθμό των διεργασιών που δίνονται. Σύμφωνα με τη μεθοδολογία, στα πρώτα στάδια πρέπει να επιλέγουμε πιο επιθετικούς διαμερισμούς.

Σαν πρώτη επιλογή κάθε διεργασία είναι υπεύθυνη για ένα κομμάτι με υποδιαιρεμένες διαστάσεις της αρχικής εικόνας. Σα να έχει δηλαδή η κάθε διεργασία μια δικιά της ξεχωριστή εικόνα. Σε αυτή την απόφαση μας οδήγησε η απόφασγ για ιδιαίτερη έμφαση στην ισοκατανομή του φορτίου ανάμεσα στις διεργασίες, εφόσον η κάθε διεργασία θα έχει τη μνήμη της όπου θα περιέχεται το αντίστοιχο κομμάτι της εικόνας.

Με αυτόν τον τρόπο:

- Έγινε εφικτή η δημιουργία εργασιών που ξεπερνούν τον αριθμό των διαθέσιμων επεξεργαστών.
- Αποφεύγεται η περιττή εκτέλεση υπολογισμών.
- Οι εργασίες έχουν ίδιο φόρτο εργασίας.
- Έχουμε αριθμό εργασιών που κλιμακώνουν με το μέγεθος του προβλήματος.
- Επιτυγχάνεται μικρή κατανάλωση μνήμης μέσω της υλοποίησης parallel I/O αφού η κάθε διεργασία διαβάζει το κομμάτι της εικόνας που της αναλογεί. Ταυτόχρονα αυτό είναι πολύ πιο γρήγορο από άποψη χρόνου.

Πιο αναλυτικά, κατά την εκτέλεση του προγράμματος, ο χρήστης επιλέγει το πλήθος των διεργασιών για το οποίο θα εκτελεστεί το πρόγραμα. Βάση ενός αλγορίθμου ο οποίος περιγράφεται στη συνάρτηση divide_rows υπολογίζεται ο βέλτιστος τρόπος με τον οποίο θα διανεμηθεί ο πίνακας στις διεργασίες.

6. Όπως ήδη προαναφέρθηκε, οι διεργασίες μοιράζονται στο πλέγμα με τέτοιο τρόπο ώστε να είναι η μία δίπλα στην άλλη σε αύξουσα σειρά. Αυτό διευκολύνει κάθε διεργασία στην ανεύρεση των γειτονικών διεργασιών της. Ιδιαίτερη περίπτωση αποτελούν οι διεργασίες που βρίσκονται στην περιφέρεια του πλέγματος. Σε αυτήν την περίπτωση γειτονικές θεωρούνται οι διεργασίες εκείνες που βρίσκονται στην κατάλληλη εκτός ορίων απέναντι πλευρά.

Προχωρώντας τώρα σε θέματα κώδικα σαν επιλογή έχουμε αποφασίσει να χρησιμοποιήσουμε nonblocking αποστολές και λήψεις μηνυμάτων μπορώντας με αυτόν τον τρόπο να κάνουμε συνέλιξη για τα κεντρικά κομμάτια του πίνακα (inner data) και αφήνοντας για αργότερα τον υπολογισμό των περιφερειακών κομματιών(outer and corner data). Αυτό δίνει την δυνατότητα σε μια διεργασία να ασχολείται με μια εργασία όσο περιμένει να έρθουν τα γειτονικά κελιά του πίνακα. Όταν ολοκληρωθεί η εργασία αυτή αν έχουν έρθει τα δεδομένα που περιμένει προχωράει στην εκτέλεση των απαιτούμενων νέων εργασιών αλλιώς εξακολουθεί να περιμένει την λήψη των απαιτούμενων δεδομένων.

Code optimization

Στην εκφώνηση της εργασίας δίνονται κάποιες ρητές και οδηγίες για την βελτιστοποίηση της υλοποίησης, οι οποίες και ακολουθήθηκαν κατά τη διεκπεραίωση της εργασίας μας. Αναλυτικά ακολουθήθηκαν οι εξής προτεινόμενες από το διδάσκοντα τεχνικές:

- Mapping – Καρτεσιανή Τοπολογία για τη διατήρηση γειτόνων
- Επικάλυψη επικοινωνίας με υπολογισμούς
- Χρήση halo points
- NULL Process
- Inline συναρτήσεις
- Χρήση datatypes
- All reduce
- Δέσμευση με malloc στους πίνακες προς αποφυγή διπλοτύπων
- Init
- Όχι αχρείαστες αποδόσεις τιμών

Έχοντας κάνει μια περιεκτική εισαγωγή αναφορικά με την εργασία, παρακάτω επισυνάπτονται ενδεικτικά **Run Times** των προγραμμάτων, συνοδευόμενα με τα κατ'αντιστοιχία υπολογισμένα **Speedup** και **Efficiency.** Σε αυτό το σημείο καλό είναι να ορίσουμε τα 2 προαναφερθέντα μεγέθη. Σαν **Speedup** για ένα πρόβλημα δεδομένου μεγέθους n ορίζουμε το λόγο του σειριακού run time προς το παράλληλο run time, δηλαδή **S = Ts/Tp,** ενώ ως Efficiency ορίζεται το πηλίκο S/p, δηλαδή **E = S/p,** όπου p ο αριθμός των processes στα οποία μοιράζεται ένα πρόγραμμα. Οι μετρήσεις είναι σε seconds και στις πράξεις για Speedup και Efficiency έγινε στρογγυλοποίηση προς τα πάνω στο 2^{o} δεκαδικό ψηφίο.

MPI Run Times without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	2.743529	5.502509	11.801090	23.600892	27.500542
4	1.592533	1.155112	1.970191	2.900897	3.002501
9	0.723983	0.657508	1.334210	1.789004	1.900847
16	0.150162	0.306655	0.877013	0.999815	1.687003
25	0.410145	0.487540	0.499010	0.489700	0.588741
36	0.616680	0.678550	1.001010	0.887521	1.000850

MPI Speedup without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	1.72	4.76	5.99	8.14	9.16
9	3.79	8.37	8.85	13.19	14.47
16	18.27	17.94	13.46	23.60	16.30
25	6.69	11.29	23.65	48.19	46.71
36	4.45	8.11	11.79	26.59	27.48

MPI Efficiency without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.43	1.19	1.50	2.03	2.29
9	0.42	0.93	0.98	1.47	1.61
16	1.14	1.12	0.84	1.48	1.02
25	0.27	0.45	0.94	1.93	1.86
36	0.12	0.23	0.47	0.74	0.76

MPI Run Times without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	9.479841	14.502396	29.178832	59.232016	120.005021
4	2.674698	7.276056	9.623111	25.124026	59.141206
9	0.554917	1.889276	3.093616	5.414209	11.938500
16	0.375939	0.692657	2.583245	3.954264	6.007254
25	0.306924	0.845311	1.400344	2.788387	4.678892
36	0.524021	1.224841	1.331055	1.988487	2.180085

MPI Speedup without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	3.54	1.99	3.03	2.36	2.01
9	17.08	7.67	9.43	10.94	10.05
16	25.22	20.94	11.30	14.98	19.98
25	30.89	17.16	20.84	21.24	25.65
36	18.09	11.84	21.92	29.79	55.05

MPI Efficiency without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.89	0.50	0.76	0.59	0.50
9	1.9	0.85	1.05	1.22	1.12
16	1.58	1.31	0.71	0.94	1.25
25	1.24	0.69	0.83	0.85	1.03
36	0.50	0.33	0.61	0.83	1.53

MPI OpenMP Run Times without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	2.520592	5.394706	11.008433	22.001121	45.008241
4	1.066115	2.054290	3.009482	5.115810	10.829458
9	0.274702	0.813776	1.650084	3.227232	6.511001
16	0.154119	0.300786	0.900810	2.008101	4.400010
25	0.047799	0.228014	0.458281	0.997841	2.100847
36	0.081743	0.137477	1.008101	2.232294	4.887610

MPI Open Mp Speedup without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	2.36	5.63	3.66	4.30	4.16
9	9.18	6.63	6.67	6.82	6.91
16	16.35	17.94	12.22	10.96	10.23
25	52.73	23.66	24.02	22.05	21.42
36	30.84	39.24	10.92	9.86	9.21

MPI OpenMP Efficiency without All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.59	1.40	0.91	1.08	1.04
9	1.02	0.74	0.74	0.76	0.77
16	1.02	1.12	0.76	0.69	0.64
25	2.1	0.95	0.96	0.88	0.86
36	0.86	1.09	0.30	0.27	0.26

MPI OpenMP Run Times without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	7.263089	15.666690	31.426049	63.085011	128.000101
4	1.980191	5.708651	9.802346	21.112841	31.448494
9	0.842134	1.055505	3.129766	6.799401	13.849542
16	0.673966	2.285249	1.936089	2.887674	5.998484
25	0.261408	0.884538	3.293088	4.211841	10.008462
36	0.520010	0.728541	4.124813	5.900841	12.811091

MPI OpenMP Speedup without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	3.67	1.60	3.21	2.99	4.07
9	8.62	14.84	10.04	9.28	9.24
16	10.78	6.86	16.23	21.85	21.34
25	27.78	17.71	9.54	14.98	12.79
36	13.97	21.50	7.62	10.69	9.99

MPI OpenMP Efficiency without All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.92	0.40	0.80	0.75	1.02
9	0.96	1.65	1.12	1.03	1.03
16	0.67	0.43	1.01	1.36	1.33
25	1.11	0.71	1.90	0.60	0.51
36	0.39	0.60	0.21	0.30	0.28

MPI Run Times with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	3.135520	6.170242	12.446750	25.010172	50.110014
4	1.219995	1.763733	3.119491	6.229001	12.448401
9	1.189013	1.086993	1.390084	2.880011	4.224411
16	0.999711	0.689979	1.000820	1.810174	3.500401
25	4.124945	6.007292	12.010072	23.990920	47.001074
36	6.339001	8.500572	17.001001	34.008171	68.110771

MPI Speedup with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	2.57	3.50	3.85	4.02	4.13
9	2.64	5.67	8.64	8.68	11.86
16	3.14	8.94	12.00	13.82	14.31
25	0.76	1.03	1.00	1.04	1.06
36	0.49	0.73	0.70	0.73	0.74

MPI Efficiency with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.64	0.87	0.96	1.00	1.03
9	0.29	0.63	0.96	0.96	1.32
16	0.19	0.56	0.75	0.86	0.89
25	0.03	0.04	0.04	0.04	0.04
36	0.02	0.02	0.09	0.02	0.02

MPI Run Times with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	7.838455	15.998741	32.001014	63.330041	26.667850
4	2.182664	4.228101	8.557890	18.000101	36.001010
9	0.669135	0.900840	2.540872	5.818170	12.078210
16	0.389900	0.590013	0.998740	1.900701	3.999871
25	0.198700	0.303304	0.637789	1.1890022	2.311011
36	0.440011	0.740054	1.490084	2.9001134	5.110149

MPI Speedup with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	3.54	3.78	3.97	3.52	3.51
9	11.71	17.75	12.59	10.88	10.48
16	20.10	26.67	32.04	33.32	31.67
25	39.45	54.45	50.17	53.26	54.51
36	17.81	21.62	21.48	21.84	24.78

MPI Efficiency with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.88	0.95	0.99	0.88	0.88
9	1.30	1.97	1.40	1.20	2.62
16	1.25	1.67	2.00	2.08	1.98
25	1.58	2.19	2.01	2.13	2.19
36	0.49	0.60	0.60	0.61	0.69

MPI OpenMP Run Times with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	3.859745	5.995662	12.100500	24.410021	49.103051
4	1.399093	1.631623	3.201151	6.190841	12.120143
9	1.101971	1.347631	2.677802	4.887154	9.000101
16	0.950044	2.881714	5.447401	10.997884	21.340456
25	0.874502	3.220740	6.313501	12.798900	25.758294
36	1.550013	4.200174	8.550115	16.990310	33.263702

MPI OpenMP Speedup with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	2.76	3.67	3.78	3.94	4.05
9	3.50	4.49	4.52	4.94	5.46
16	4.06	2.08	2.22	2.21	2.30
25	4.41	1.86	1.92	1.91	1.90
36	2.44	1.43	1.42	1.44	1.47

MPI_OpenMP Efficiency with All reduce(Grey)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.69	0.92	0.95	0.99	1.01
9	0.39	0.50	0.50	0.55	0.61
16	0.25	0.13	0.14	0.14	0.14
25	0.18	0.07	0.08	0.08	0.08
36	0.07	0.04	0.04	0.04	0.04

MPI OpenMP Run Times with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	8.007805	16.601732	33.033100	65.030500	131.131001
4	2.712185	6.579014	13.017312	26.221200	52.725401
9	0.927491	1.894525	3.450400	9.551301	20.001321
16	0.358354	1.693135	2.930044	6.110154	13.123245
25	1.951205	3.567109	7.113256	14.521330	29.100101
36	2.400121	4.991300	9.788010	19.001810	38.210550

MPI OpenMP Speedup with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	2.96	2.52	2.54	2.48	2.49
9	8.63	8.76	9.58	6.81	6.56
16	2.27	9.81	11.27	10.64	9.99
25	4.57	4.65	4.64	4.48	4.51
36	3.34	3.33	3.37	3.42	3.43

MPI OpenMP Efficiency with All reduce(RGB)

Processes	960*1260	1920*1260	1920*2520	3840*2520	3840*5040
1	1	1	1	1	1
4	0.74	0,63	0.64	0.62	0.62
9	0.96	0,97	1.06	0.76	0.73
16	0.14	0.61	0.70	0.67	0.62
25	0.18	0.19	0.19	0.18	0.18
36	0.09	0.09	0.09	0.09	0.09

Συμπερασματικά

Η συγκεκριμένη εργασία όπως φαίνεται και από τις άνωθι μετρήσεις, εκτός από το προγραμματιστικό της ενδιαφέρον ως προς την υλοποίηση του κώδικα, έχει και σημαντικό ερευνητικό ενδιαφέρον, καθώς μέσα από τις μετρήσεις που κληθήκαμε να κάνουμε προέκυψαν κάποια συμπεράσματα τα οποία και κρίνεται σκόπιμο να τονίσουμε.

Αρχικά, όπως είναι φυσιολογικό και θα υπέθετε κανείς, βγαίνει το συμπέρασμα ότι για δεδομένο μέγεθος προβλήματος , όσο αυξάνονται οι διεργασίες μειώνεται το run time και συνεπακόλουθα αυξάνεται και η επιτάχυνση και άρα η αποτελεσματικότητα. Αυτή ωστόσο η αύξηση της αποτελεσματικότητας κάποια στιγμή όσο αυξάνουμε τα processes, πάντα για δεδομένο μέγεθος προβλήματος, φτάνει ένα σημείο καμπής και αρχίζει να μειώνεται. Αυτό προκύπτει φυσιολογικά, καθώς όπως μάθαμε και στη θεωρία, όσο αυξάνονται οι πυρήνες τόσο αυξάνεται και το κόστος επικοινωνίας μεταξύ των πυρήνων αλλά και το κόστος αμοιβαίου αποκλεισμού λόγω κοινών πόρων. Καταλήγει λοιπόν αυτό το κόστος να υπερκερνά τη βελτίωση που προσφέρει ο παραλληλισμός στη λύση ενός συγκεκριμένου προβλήματος και καταλήγουμε να παίρνουμε χειρότερα αποτελέσματα. Θα μπορούσαμε να φανταστούμε το εξής σενάριο προκειμένου να αποδώσουμε καλύτερα το παραπάνω συμπέρασμα : έστω μια οικοδομή 5 ορόφων και 5 εργάτες που δουλεύουν σε αυτήν. Εαν βάλουμε 5 εργάτες ακόμα, θα αυξηθεί η δυνατότητα διεκπαιρέωσης του έργου σημαντικά με κάποιο μικρό κόστος λόγω της ανάγκης συντονισμού τους. Εάν βάλουμε 50 επιπλέον εργάτες, αυξάνεται κι άλλο η δυνατότητα διεκπαιρέωσης του έργου, αλλά η οργάνωση και η επικοινωνία μεταξύ των εργατών για την ομαλή περάτωση των εργασιών γίνεται σε πολύ μεγαλύτερο βαθμό πιο δύσκολη. Όταν λοιπόν φτάσουμε στο κρίσιμο σημείο όπου πλέον το κόστος επικοινωνίας μεταξύ των εργατών φτάνει να είναι πιο μεγάλο από τη βελτίωση που εκείνοι προσφέρουν στη γρηγορότερη ολοκλήρωση του οικοδομήματος, τότε όσο συνεχίζουμε να προσθέτουμε εργάτες η αποδοτικότητα θα γίνεται με όλο και μεγαλύτερο ρυθμό χειρότερη. Βεβαίως, πρέπει να τονιστεί ότι εαν αλλάζει ο το μέγεθος του προβλήματος, αλλάζει και αυτό το σημείο

που το Toverhead θα γίνεται πιο σημαντικό από τη βελτίωση που προσφέρει η παραλληλία, για παράδειγμα αν μεγαλώνει το μέγεθος ενός προβλήματος, αντίστοιχα μεγαλώνει και ο αριθμός των πυρήνων για τον οποίο το tradeoff που γίνεται είναι για μας αρνητικό.

Στην εργασία το κόστος της επικοινωνίας μεταξύ των πυρήνων γίνεται ολοφάνερο στην υλοποίηση με all reduce, όπου οι διεργασίες πρέπει μετά το πέρας κάθε γενιάς να συγχρονίζονται. Στις μετρήσεις μας το κόστος αυτό φανερώνει ξεκάθαρα την αρνητική πλευρά του παραλληλισμού, καθώς το κόστος της επικοινωνίας εν προκειμένω είναι ιδιαίτερα μεγάλο και οι αποδοτικότητες που μετρήσαμε ήταν χειρότερες κι από τη σειριακή λειτουργία αρκετές φορές.

Παρ’ όλα αυτά, το ευρύ συμπέρασμα είναι ότι υπό τις προϋποθέσεις που αναλύσαμε παραπάνω, η παραλληλία μπορεί να κάνει τη λύση ενός προβλήματος αποδοτικότερη, αρκεί να μην διαμοιράζεται η επίλυσή του σε υπερβολικό αριθμό διεργασιών σχετικά με το μέγεθός του. Μάλιστα συμπεράναμε ότι πολλές φορές το speedup ήταν superlinear και η αποδοτικότητα εντυπωσιακή, πριν να αρχίσει να πέφτει λόγω της υπερβολικής αύξησης των διεργασιών σε σχέση με το δοσμένο μέγεθος προβλήματος. Κλείνοντας, αν μπορούμε να πούμε κάτι με περίσσεια σιγουριά, είναι ότι σε αντίθεση με το νόμο του Amdahl, ο οποίος υποτιμά την παραλληλία και δε λαμβάνει υπόψην το μέγεθος ενός προβλήματος, είναι ότι η σχέση μεταξύ μεγέθους προβλήματος και ανάθεσης σε διεργασίες είναι άρρηκτη και με τη σωστή της χρήση τα αποτελέσματα που επιτυγχάνονται είναι εντυπωσιακά.