Parallel Systems
Heat distribution

1115201300143 Μανώλης Πιτσικάλης, 1115201300195 Παναγιώτης Φωτόπουλος

8 Οκτωβρίου 2016

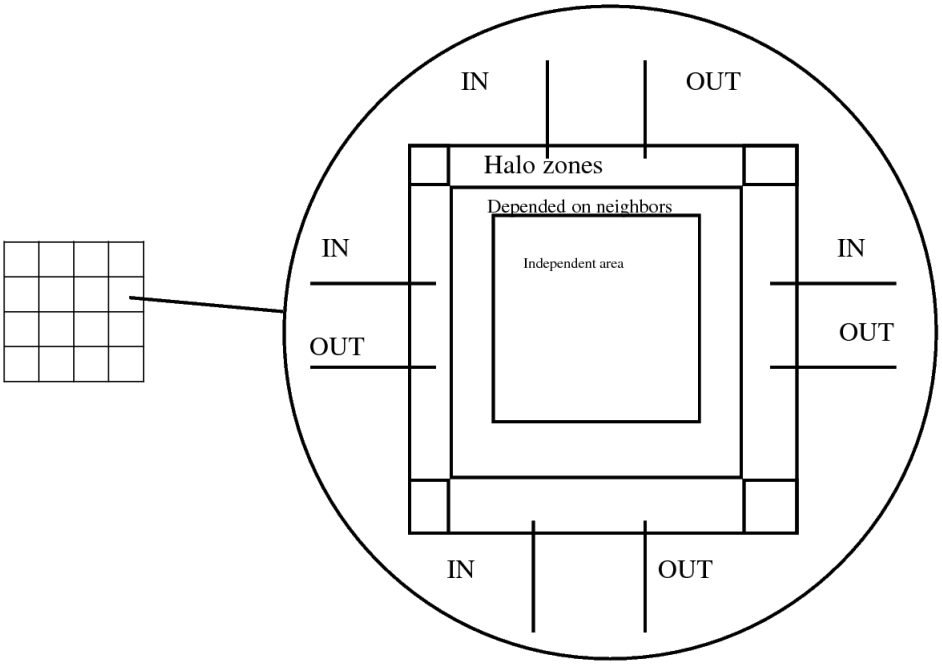
1. Αρχεια
   -Makefile
   -mpi_heat_improved_persistent_stat.c
   Με την εντολη make παράγονται τα αρχεια:
   –heat_size εκτέλεση προβλήματος μεγέθους size και βημάτων x
   –heat_con_size εκτέλεση προβλήματος μεγέθους size και βημάτων x και έλεγχος σύγκλισης
   –heat_omp_size εκτέλεση προβλήματος μεγέθους size και βημάτων x μαζί με openmp
   –heat_con_omp_size εκτέλεση προβλήματος μεγέθους size και βημάτων x μαζί με openmp και έλεγχο σύγκλισης
   Οι διάφορες παραμέτροι ρυθμίζονται από το Makefile.

2. MPI
   Για το MPI έγιναν οι εξής οι βελτιώσεις και σχεδιαστικές επιλογές:

   (α′) Τοπολογία
   Για την τοπολογία επιλέχθηκε διαμοιρασμός σε blocks, συγκεκριμένα χρησιμοποιήθηκε η συνάρτηση MPI_Cart_create για την δημιουργία καρτεσιανής τοπολογίας.

   (β′) Block
   Κάθε block αναλάμβανει μια περιοχή της μεταλλικής πλάκας. Οι περιοχές με στοιχεία που συνορεύουν με άλλα blocks χρειάζονται για τον υπολογισμό τιμής τους τιμές από τα δεύτερα επομένως εμφανίζεται η αναγκή για επικοινωνία μεταξύ τους. Για την επίλυση αυτού του ζητήματος χρησιμοποιούνται τέσσερεις halo zones(βορράς, νότος, ανατολή, δύση) στις οποίες αποθηκεύονται τα στοιχεία που χρειάζονται από τα γειτονικά blocks αντίστοιχα.

Σχήμα 1: Table 1



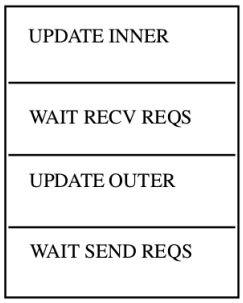
(γ′) Πίνακας block
Κάθε block έχει δικό του κομμάτι σε ένα πίνακα μεγέθους ίσο με (μέγεθος προβλήματος +2)*(μέγεθος προβλήματος +2) έτσι ώστε να υπάρχει σε κάθε περίπτωση χώρος για halo zones , επίσης για να διατηρείται ο παλιός και ο καινούριος πίνακας χρησιμοποιούνται δύο τέτοιοι πίνακες οι οποίοι σε κάθε επανάληψη αλλάζουν ρόλο, επιπλεόν οι πίνακες είναι στατικόι α΄ρα δεν έχει το overhead ενός δυναμικά δεσμευμένου.

(δ′) Datatypes
Ορίστηκαν τα εξής MPI Datatypes :
-row_type contiguous
-col_type vector
Για την αποστολή των αρχικών/τελικών δεδομένων από/προς τον master χρησιμοποιείται row_type ενώ για την διαδικασία της αποστολής μηνυμάτων μεταξύ blocks χρησιμοποιούνται row_type για τον βορρά και τον νότο και col_type για ανατολή και δύση.

(ε′) Αναμονή εξωτερικών στοιχείων
Κατά την διάρκεια αναμονής των στοιχείων ενημερώνονται τα εσω-

τερικά ανεξάρτητα στοιχεία του block έτσι μειώνεται ο χρόνος στον οποίο η διεργασία δεν εκτελεί κάποια δουλειά.

Σχήμα 2: Table 1



(ϛ′) Αποστολή μηνυμάτων
Για να μειωθεί το overhead των send/receive μιας και οι γείτονες δεν αλλάζουν χρησιμοποιούνται persistent send/receive, επιπλεόν αν καποίος γείτωνας δεν υπάρχει το id του είναι -1 άρα δεν χρειάζονται έλεγχοι γι′ αυτη την περίπτωση .

(ζ′) Γενικά
Αποφύγαμε χρήση συναρτήσων και τοπικών μεταβλητων για καλύτερη απόδοση.

(η′) Σύγκλιση
Για τον έλεγχο σύγκλισης επιλέγθηκε να γίνεται κάθε 20 βήματα και η διαφο′ρα των στοιξχείων να είναι μικρότερη του $1e{-}3$.Αναλυτικότερα κάθε διεργασία αφού υπολογίσει τα εξωτερίκα στοιχεία και πριν εκπληήρωσει τα send request ελέγχει κάθε στοιχείο του νέου πίνακα της σε σχέση με τον παλιό

3. Openmp
Η παραλληλία με χρήση openmp έγινε στην ενημέρωση του εσωτερικού ανεξάρτητου τμήματος του πίνακα και στο εξωτερικό εξαρτημένο (for loops). Στις μετρήσεις χρησιμοποιήθηκαν 4 threads.

4. CUDA
Η υλοποίηση σε Cuda έγινε με τις εξης επιλογές - λεπτομέρειες:

(α′) Αριθμός των μπλοκ και των νημάτων
Ο αριθμός των νημάτων παραμένει σταθερός, και ο αριθμός των

μπλοκ προσαρμόζεται αναλογα με τον αριθμό των νημάτων και το μέγεθος του προβλήματος. Όπως φαίνεται και στις μετρήσεις, οι εναλλακτικές τιμές για τον αριθμό των νημάτων είναι οι 4, 8, 16, 32 (διαιρέτες του 32, προς επίτευξη καλύτερης απόδοσης, καθ΄ω το μέγεθος και γενικότερα η σωστή ξρήση του warp παίζει μεγάλο ρόλο στην απόδοση.

(β΄) Χωρισμός του πίνακα
Κάθε μπλοκ παίρνει ενα κομμάτι του πίνακα, μεγέθους ίσου με τον αριθμό των νημάτων του. Σε κάθε νήμα αντιστοιξεί ένα στοιχείο του πίνακα. Ο πίνακας φορτώνεται ολόκληρος στην μνήμη της GPU, οπότε όλα τα μπλοκς βλέπουν όλο τον πίνακα και απλώς επεξεργάζονται τα στοιχεία που τα αφορούν. Νήματα τα οποία βρισκονται τυχόν εκτός πίνακα (λόγω πιθανής ασσυμετρίας του μέγεθους του προβλήματος με τον αριθμό των νημάτων ανά μπλοκ) παραμένουν σε κατάσταση αναμονής (idle), ώστε να μην επηρεάζουν το υπόλοιπο πρόγραμμα.

(γ΄) Γενικότερες παρατηρήσεις
Έγιναν προσπάθειες για βελτίωση της απόδοσης, όπως κατά τα πρώτα βήματα του ελέγχου σύγκλισης έγινε ξεδίπλωμα των επαναλήψεων (για βελτίωση ταχύτητας, ξρήσης μνήμης κλπ), περιοροσμός στο ελάχιστο των αναγνώσεων/εγγραφών στην/από τη μνήμη της GPU, και άλλες μικρότερες λεπτομέρειες και στοιχεία του κώδικα τα οποία συνολικά βελτίωσαν το αποτέλεσμα.

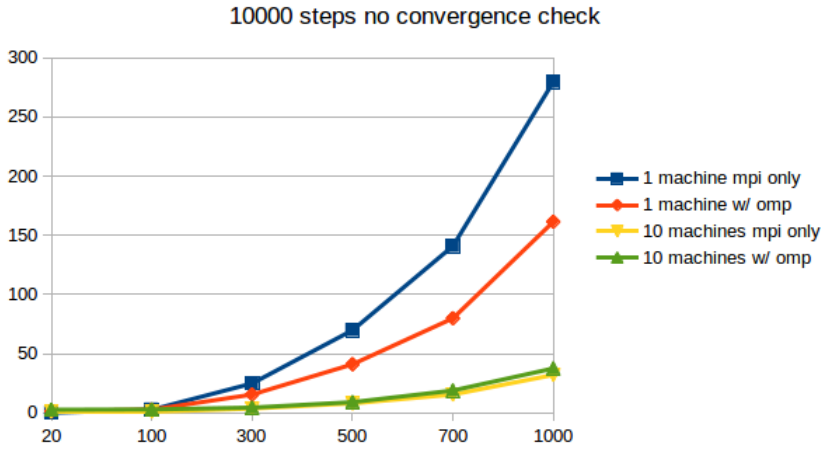
(δ΄) Σύγκλιση
Επιλέχθηκαν οι ίδιες παράμετροι όπως στο MPI. Κάθε μπλοκ ελέγξει εσωτερικά αν όλα τα στοιχεία του έξουν συγκλίνει, και έπειτα η διαδικασία επαναλαμβάνεται για να ελεγθεί συνολικά ο πίνακας.

5. Μετρήσεις mpi

Οι μετρήσεις έγιναν στα μηχανήματα του εργαστηρίου linux για το mpi και στο cluster για το cuda, και δεν είναι σίγουρα αντιπροσωπευτικά καθώς μπορεί να εκτελούταν και κάποιο άλλο πρόγραμμα.Παρ όλα αυτά τα αποτελέσματα σχολιάζονται και στις περισσότερες περιπτώσεις ήταν τα αναμενόμενα.

Πίνακας 1: 10000 steps no convergence

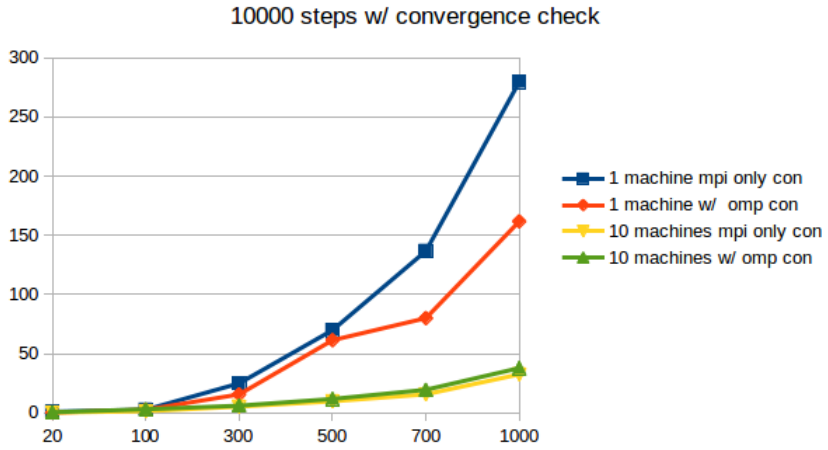
	20	100	300	500	700	1000
1 machine mpi only	0.1	2.7	24.9	69.9	140.96	279.68
1 machine w/ omp	1.3	2.8	15.5	41.1	79.91	161.47
10 machines mpi only	1.2	1.2	3.7	8.03	15.5	32.1
10 machines w/ omp	2.7	3	4.5	9.13	18.94	37.6
Efficiency mpi	0.008	0.225	0.673	0.870	0.909	0.871
Efficiency mpi w/omp	0.048	0.093	0.344	0.450	0.422	0.429
Speedup mpi	0.083	2.250	6.730	8.705	9.094	8.713
Speedup mpi w/omp	0.481	0.933	3.444	4.502	4.219	4.294



Η χρήση ομαδικής επικοινώνιας για τον έλεγχο σύγκλισης επιδρά αρνητικά στους χρονούς κάτι που ήταν αναμενόμενο αφού η επικοινωνία έχει κόστος. Τα αποτελέσματα στον παρακάτω πινακα.

Πίνακας 2: 10000 steps w/ convergence

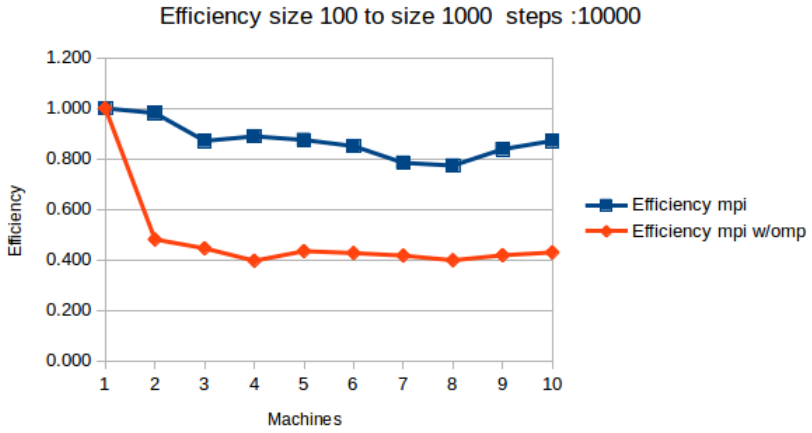
	20	100	300	500	700	1000
1 machine mpi only con	1	2.73	24.97	69.94	136.79	279.68
1 machine w/ omp con	0.24	2.7	15.66	61.4	80.03	161.79
10 machines mpi only con	0.27	1.46	5.3	9.8	15.69	32.6
10 machines w/ omp con	0.55	3.1	6.1	11.8	19.54	37.9
Efficiency mpi	0.370	0.187	0.471	0.714	0.872	0.858
Efficiency mpi w/omp	0.044	0.087	0.257	0.520	0.410	0.427
Speedup mpi	3.704	1.870	4.711	7.137	8.718	8.579
Speedup mpi w/omp	0.436	0.871	2.567	5.203	4.096	4.269



Στον παρακάτω πίνακα φαίνεται πως η αποτελεσμάτικοτητα μένει σχετικά σταθερή όσο αυξάνεται το μέγεθος του προβλήματος και αντίστοιχα το πλήθος των διεργασιών, έτσι φαινεται να έχει καλη κλιμάκωση, βέβαια φαίνεται ότι το πρόγραμμα με openmp έχει χαμηλότερη αποτελεσματικότητα και χειρότερους χρόνους.

(α′) 100 to 1000 steps 10000

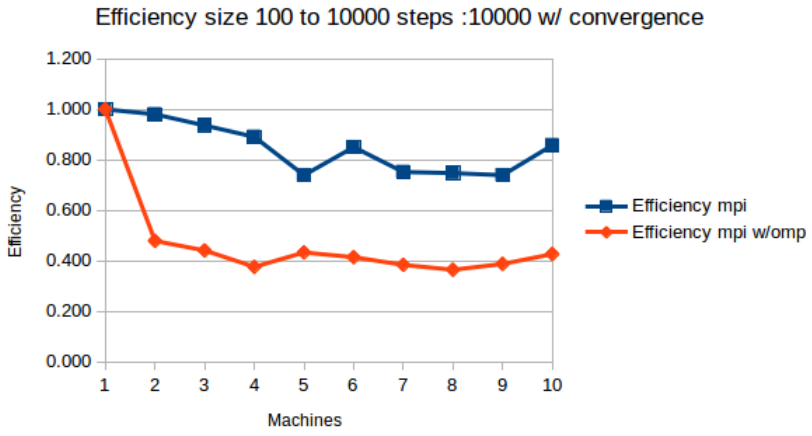
Machines	1	2	3	4	5	6	7	8	9	10
Size	100	200	300	400	500	600	700	800	900	1000
1 machine mpi only	2.7	10.99	24.9	44.63	69.8	100.47	136.71	179.41	226	279.68
1 machine w/ omp	2.8	7.5	15.56	27.04	41.31	59.52	80.12	108.31	131.73	161.47
x machines mpi only	2.7	5.6	9.53	12.546	15.96	19.69	24.9	29	29.96	32.1
x machines w/ omp	2.8	7.8	11.64	17.05	19.02	23.24	27.47	34	35	37.6
Efficiency mpi	1.000	0.981	0.871	0.889	0.875	0.850	0.784	0.773	0.838	0.871
Efficiency mpi w/omp	1.000	0.481	0.446	0.396	0.434	0.427	0.417	0.398	0.418	0.429
Speedup mpi	1.000	1.963	2.613	3.557	4.373	5.103	5.490	6.187	7.543	8.713
Speedup mpi w/omp	1.000	0.962	1.337	1.586	2.172	2.561	2.917	3.186	3.764	4.294



Παρακάτω τα αποτελέσματα των εκτελέσεων με έλεγχο σύγκλισης , οπού φαίνεται το κόστος που έχει στο efficiency η ομαδική επικοινωνία.

(α΄) 100 to 1000 steps 10000 w/ convergence

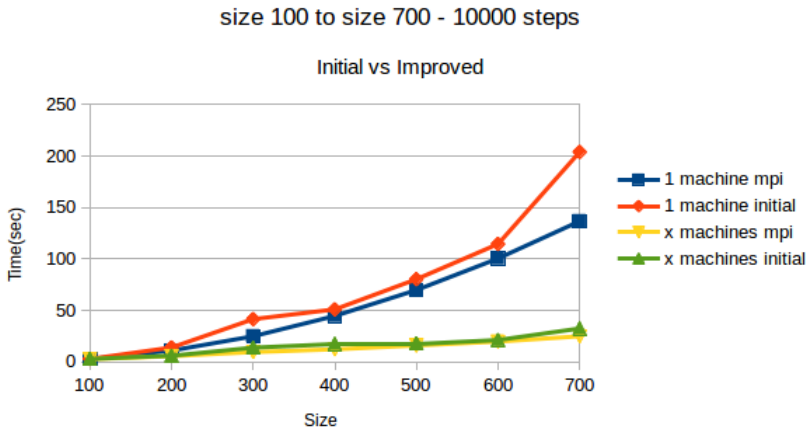
Machines	1	2	3	4	5	6	7	8	9	10
Size	100	200	300	400	500	600	700	800	900	1000
1 machine mpi only con	2.7	10.98	25	44.69	69.8	100.47	136.84	179.41	226	279.68
1 machine w/ omp con	2.7	7.54	15.56	26	41.66	59.26	80.63	110.31	132.63	161.79
x machines mpi only con	2.7	5.6	8.9	12.54	18.88	19.69	26	30	33.96	32.6
x machines w/ omp con	2.7	7.87	11.76	17.28	19.24	23.83	30	37.8	38	37.9
Efficiency mpi	1.000	0.980	0.936	0.891	0.739	0.850	0.752	0.748	0.739	0.858
Efficiency mpi w/omp	1.000	0.479	0.441	0.376	0.433	0.414	0.384	0.365	0.388	0.427
Speedup mpi	1.000	1.961	2.809	3.564	3.697	5.103	5.263	5.980	6.655	8.579
Speedup mpi w/omp	1.000	0.958	1.323	1.505	2.165	2.487	2.688	2.918	3.490	4.269



6. Σύγκριση αρχίκου=βελτιωμένου προγράμματος
   Από τα αποτελέσματα φαινεται πως το αρχίκο πρόγραμμα κάνει μεγαλυτερους χρόνους απο το βελτιωμένο, ωστόσο η αποτελεσματικότητα έχει καλύτερες τιμές.

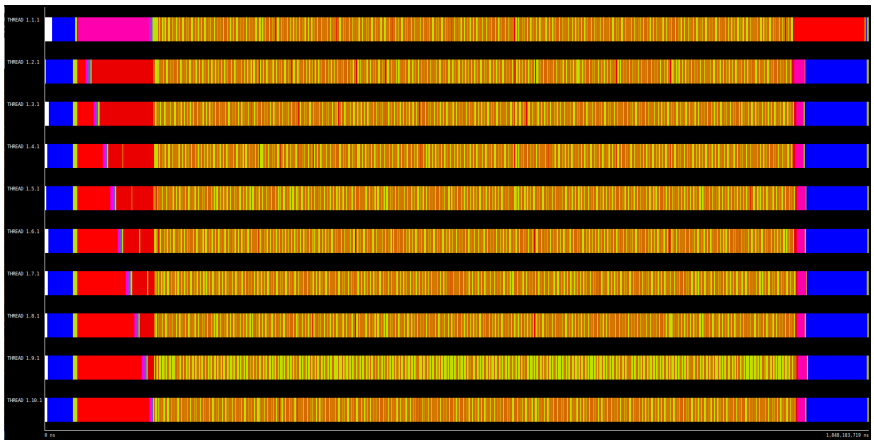
Πίνακας 5: 100 to 700 steps 10000

Machines	1	2	3	4	5	6	7
Size	100	200	300	400	500	600	700
1 machine mpi	2.7	10.99	24.9	44.63	69.8	100.47	136.71
1 machine initial	2.9	14	41.56	51.04	80.34	114.52	204
x machines mpi	2.7	5.6	9.53	12.24	15.96	19.69	24.9
x machines initial	2.9	6	13.84	17.5	18.02	21.24	32.47
Efficiency mpi	1.000	0.981	0.871	0.912	0.875	0.850	0.784
Efficiency mpi initial	1.000	1.167	1.001	0.729	0.892	0.899	0.898
Speedup mpi	1.000	1.963	2.613	3.646	4.373	5.103	5.490
Speedup mpi initial	1.000	2.333	3.003	2.917	4.458	5.392	6.283



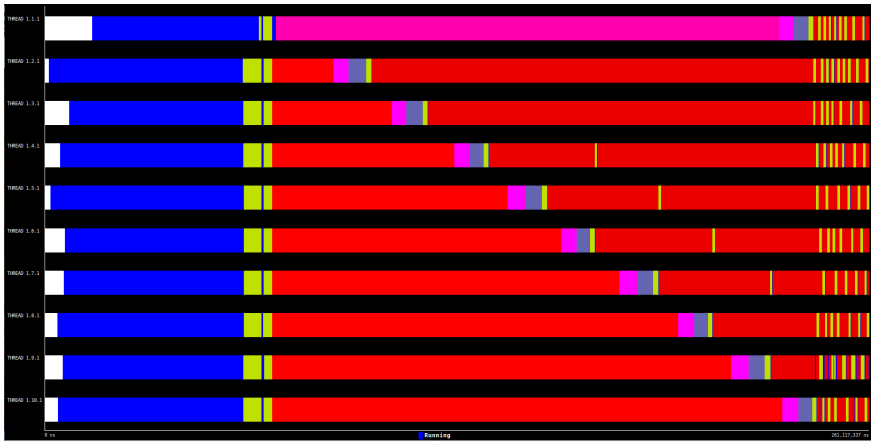
7. Paraver 300 size 500 steps
   Οι μετρησείς έγιναν με τέτοιους αριθμούς ωστέ τα αρχεία που παράγονταν να χώρανε στο quota των μηχανημάτων του εργαστηριού επιπλεόν δεν είναι σίγουρο οτι εγίναν την ώρα που δεν εκτελουταν κάποιο άλλο πρόγραμμα.

Σχήμα 3: size 300 full



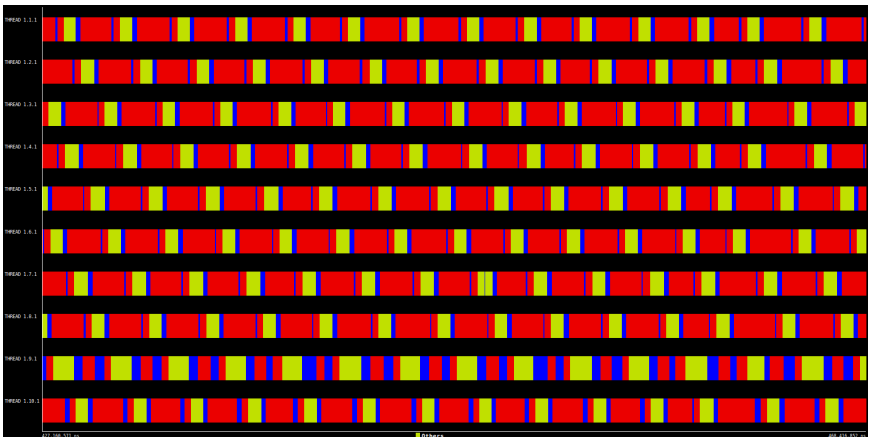
Στην παρακάτω εικόνα φαίνεται η αποστλή δεδομένων απο τον master στους υπολοιπους, γι αυτο και υπάρχει μεγάλη ποσότητα ροζ (blocking send) και αντίστοιχα κόκκινο στους υπόλοιπους.

Σχήμα 4: size 300 start



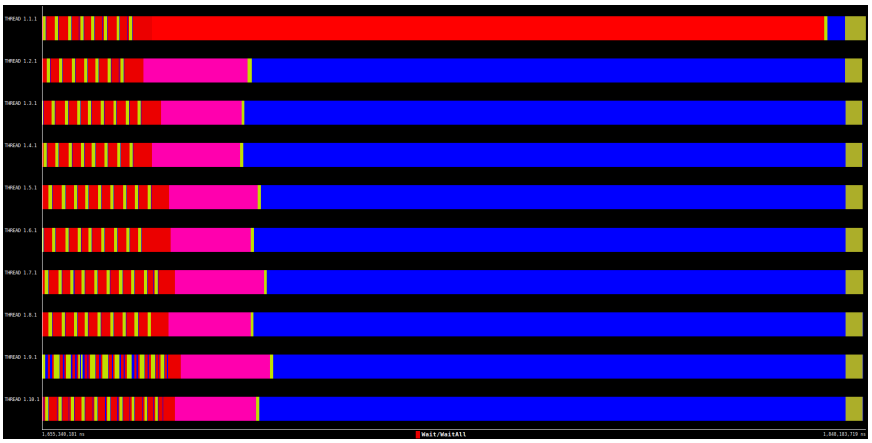
Στην παρακάτω εικόνα φαίνεται η αποστλή δεδομένων μεταξύ των διεργασιών και η εναλλαγή σε εκτέλεση από blocked, φαίνεται πως η επικοινωνια κοστίζει στο συνολικό πρόγραμμα!

Σχήμα 5: size 300 mid



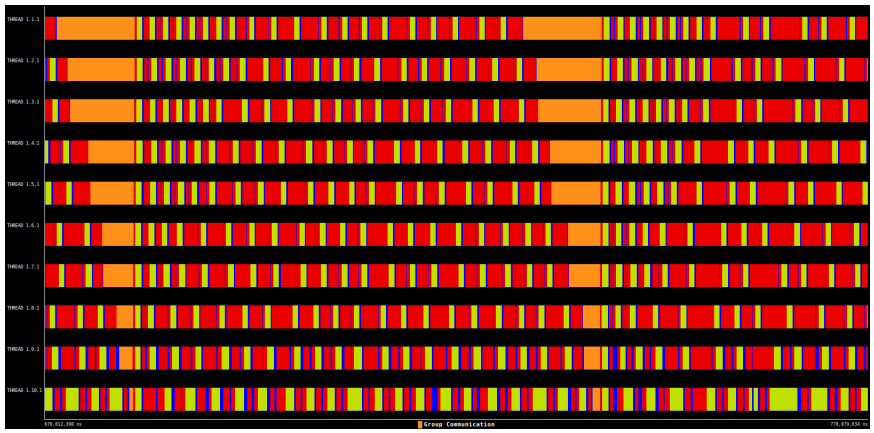
Εδώ φαινέται το τέλος του προγράμματος στο οποίο οι διεργασίες στέλνουν τα δεδομένα τους στον master, με ροζ οι αποστολές και με κόκκινο η αναμονή του master.

Σχήμα 6: size 300 end



Αξιζεί επίσης να δει κανείς πως η επικοινώνια μεταξύ ολής της ομάδας έχει τις εξέις καθυστερήσεις στην παρακάτω εικόνα (mpi_Allreduce εκτέλεση ελέγχου σύγκλισης 500 βήματα μέγεθος 300).

Σχήμα 7: size 300 mid convergence check



8. Μετρήσεις CUDA
   Αριθμός νημάτων ανά μπλοκ (κάθετα) - Μέγεθος πίνακα (μία διάσταση) (οριζόντια)

Πίνακας 6: 10000 steps no convergence

Number of threads - Size	20	100	300	500	700	1000
4	22.974ms	62.87ms	413.75ms	1.234s	2.373s	4.968s
8	21.576ms	46.36ms	223.16ms	701.89ms	1.249s	2.812s
16	25.78ms	59.28ms	318.05ms	1.094s	1.754s	4.528s
32	38.54ms	111.74ms	650.01ms	2.048s	3.471s	8.8525s

Παρατηρούμε ότι υπάρξει σημαντικότατη βελτίωση σε σχέση με το απλό mpi , καθώς και πολύ καλύτερη κλιμάκωση (συγκρίνοντας χρόνους σε μέγεθος 20 και μέγεθος 1000, σε mpi και cuda αντίστοιχα).

Πίνακας 7: 10000 steps w/ convergence

Number of threads - Size	20	100	300	500	700	1000
4	7.607ms	244.60ms	1.893s	5.402s	10.608s	21.796s
8	8.078ms	91.234ms	499.42ms	1.421s	2.673s	5.580s
16	65.734ms	116.824ms	530.957ms	1.629s	2.76s	7.580s
32	150.11ms	355.07ms	1.751s	4.941s	8.744s	20.508s

Ο έλεγχος σύγκλισης όπως ήταν αναμενόμενο προκαλεί κλιμακωτά με-

γαλύτερους χρόνους, αν και στις περιπτώσεις που έχουμε σύγκλιση τα αποτελέσματα είναι σημαντικά πιο χαμηλά, καθώς το (συξνά όχι μικρό) τελευταίο κομμάτι των επαναλήψεων δεν γίνεται.