Συστήματα Παράλληλης Επεξεργασίας

Εργαστηριακή αναφορά 6
Παραλληλοποίηση και βελτιστοποίηση αλγορίθμων σε αρχιτεκτονικές κατανεμημένης μνήμης

Γιάννος Παράνομος - 03118021
Νικήτας Τσίννας - 03118187
Καούκης Γιώργος - 03119006

Φεβρουάριος 2023

1 Εισαγωγή

Για την επίλυση του προβλήματος της διάδοσης θερμότητας σε δύο διαστάσεις, χρησιμοποιούνται τρεις υπολογιστικοί πυρήνες, οι οποίοι αποτελούν ευρέως διαδεδομένη δομική μονάδα για την επίλυση μερικών διαφορικών εξισώσεων: η μέθοδος Jacobi, η μέθοδος Gauss-Seidel με Successive Over-Relaxation (SOR) και η μέθοδος Red-Black με SOR, που πραγματοποιεί Red-Black ordering στα στοιχεία του υπολογιστικού χωρίου και συνδυάζει τις δύο προηγούμενες μεθόδους.

Λαμβάνουμε μετρήσεις για σταθερό αριθμό επαναλήψεων καθώς και με έλεγχο σύγκλισης.

2 Υλοποίηση

Έγινε υλοποίηση των 3 μεθόδων. Τα αρχεία με τους κώδικες υπάρχουν συννημένα μαζί με την εργαστηριακή αναφορά.

2.1 Jacobi

Σε αυτή την μέθοδο υπολογισμού, για να υπολογίσουμε την μεταβλητή μίας επόμενης χρονικής στιγμής χρειαζόμαστε μόνο τις μεταβλητές προηγούμενων χρονικών στιγμών.

Ακολουθήσαμε τα ακόλουθα βήματα στην υλοποίηση του Jacobi MPI:

1. Για το κάθε current στοιχείο του πίνακα, χρειαζόμαστε τα συνοριακά στοιχεία previous των θέσεων South, West, East, North. Για αυτό χρησιμοποιούμε δύο πίνακες (previous και current).

2. Προσθέτουμε row/col σε κάθε πλευρά ώστε να μπορέσουμε να μεταφέρουμε τα στοιχεία από τις γειτονικές διεργασίες.

3. Χρησιμοποιούμε την Scatter ώστε κάθε διεργασία να έχει μία local έκδοση του global πίνακα.

4. Για να μπορούμε να μεταφέρουμε μία στήλη του πίνακα με μία κλήση Send ή Recv ορίζουμε ένα datatype column.

5. Χρησιμοποιούμε την MPI_Cart_Shift για να εντοπίσουμε τις γειτονικές διεργασίες.

6. Κάνουμε επικοινωνία async για να μεταφέρουμε τα συνοριακά στοιχεία του previous στα γειτονικά με την χρήση MPI_Isend και MPI_Irecv. Κάθε επικοινωνία χαρακτηρίζεται από ένα MPI_Request. Για τον συγχρονισμό των διεργασιών χρησιμοποιούμε την MPI_Waitall.

7. Εκτελούμε τον υπολογισμό. Έπειτα μέσω ενός MPI_Barrier είμαστε σίγουροι για την ολοκλήρωση όλων των υπολογισμών.

8. Επαναλαμβάνουμε για όλες τις εποχές μέχρι την σύγκλιση.

9. Χρησιμοποιούμε την MPI_Gather για να συγκροτήσουμε τον τελικό πίνακα στην διεργασία με rank 0 η οποία τυπώνει το αποτέλεσμα.

10. Για να υπολογίσουμε τους χρόνους εκτέλεσης, χρησιμοποιούμε την MPI_Reduce από τις τοπικές μεταβλητές που αντιστοιχούν στους χρόνους (total, computational, convergence) στις αντίστοιχες global και έπειτα κρατάμε την μεγαλύτερη σε κάθε περίπτωση.

11. Για να αποφανθούμε αν συγκλίνουν όλες οι διεργασίες, γίνεται MPI_Allreduce στην global_converged από τις τοπικές converged μεταβλητές κρατώντας την μικρότερη από αυτές.

2.2 Gauss-Seidel SOR

Σε αυτή την μέθοδο υπολογισμού, αντί να χρησιμοποιούμε μόνο παρελθοντικές τιμές για την εύρεση των νέων, χρησιμοποιούμε και αντίστοιχες νέες μεατβλητές αν είναι διαθέσιμες. Στην δική μας περίπτωση, για τον υπολογισμό του

$$U[t+1][x][y]$$

, μπορούμε να χρησιμοποιήσουμε τα

$$U[t+1][x-1][y]$$

και

$$U[t+1][x][y-1]$$

. Έτσι, μεγαλώνει ο ρυθμός σύγκλισης σε σχέση με την προηγούμενη μέθοδο Jacobi καθώς η διαδικασία υπολογισμού επιταγχύνεται.

Γενικά, η υλοποίηση του Gauss-Seidel μοιάζει με αυτή του Jacobi, ωστόσο αλλάζει ο τρόπος επικοινωνίας. Ποιο συγκεκριμένα, το κάθε στοιχείο χρειάζεται να έχει το current των North, West και το previous των South, East. Αντίστοιχα, στέλνει το δικό του previous στα North και West και το current, μετά τον υπολογισμό του, στα South, East.

Σε αυτή την υλοποίηση υποβαθμίζεται ο παραλληλισμός, καθώς το κάθε thread πρέπει να περιμένει τα πάνω και δεξιά του threads για να ξεκινήσει τον υπολογισμό. Όμως "κερδίζουμε" στην ταχύτητα της σύγκλισης, η οποία είναι μεγαλύτερη σε σχέση με την Jacobi μέθοδο.

2.3 Redblack SOR

Η μέθοδος αυτή βασίζεται στην Gauss-Seidel SOR με την διαφορά ότι χωρίζει τα στοιχεία σε δύο ομάδες (κόκκινη και μαύρη). Η ανανέωση γίνεται σε δύο φάσεις (στην κόκκινη φάση υπολογίζονται τα κόκκινα από τα μαύρα, ενώ στην μάυρη φάση υπολογίζονται τα μαύρα από τα κόκκινα). Έτσι βελτιώνουμε την ταχύτητα σύγκλισης αφού αντιμετωπίζουμε το πρόβλημα της αναμονής.

Για την υλοποίηση σκεφτόμαστε τις δύο φάσεις του αλγορίθμου όπως αναφέρθηκαν προηγουμένως. Δηλαδή, εφαρμόζουμε επικοινωνία send-receive για τα previous στοιχεία όλων των γειτονικών για να υπολογίσουμε τα κόκκινα. Έπειτα, υπολογίζουμε τα μαύρα επικοινωνώντας με όλα τα γειτονικά στοιχεία για να τους στείλουμε τα current.

3 Μετρήσεις με έλεγχο σύγκλισης

Παρουσιάζονται τα αποτελέσματα για τον έλεγχο σύγκλισης. Οι μετρήσεις πραγματοποιήθηκαν για μέγεθος πίνακα 1024x1024 και 64MPI διεργασίες.

Μέθοδος	Total Time	Computation Time	Midpoint
Jacobi	535.42	41.22	5.43
Gauss-Seidel SOR	13.21	0.61	5.64
Redblack SOR	2.50	0.31	5.64

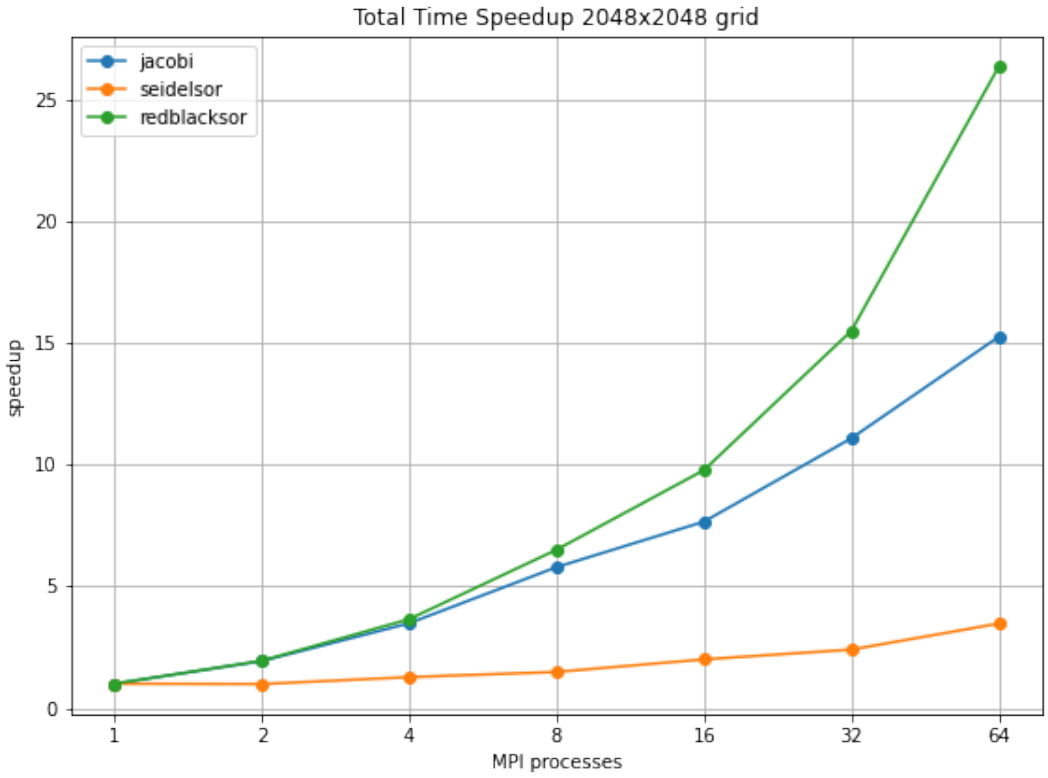
Παρατηρούμε πως η μέθοδος Jacobi έχει την χειρότερη επίδοση καθώς καθυστερεί κατά περίπου 40 φορές περισσότερο από την GaussSeidel και 200 από την RedBlack. Αυτό το περιμέναμε, καθώς όπως εξηγήσαμε και παραπάνω, η Jacobi, δεν αξιοποιεί τις ήδη υπολογισμένες τιμές της τωρινής χρονικής στιγμής.

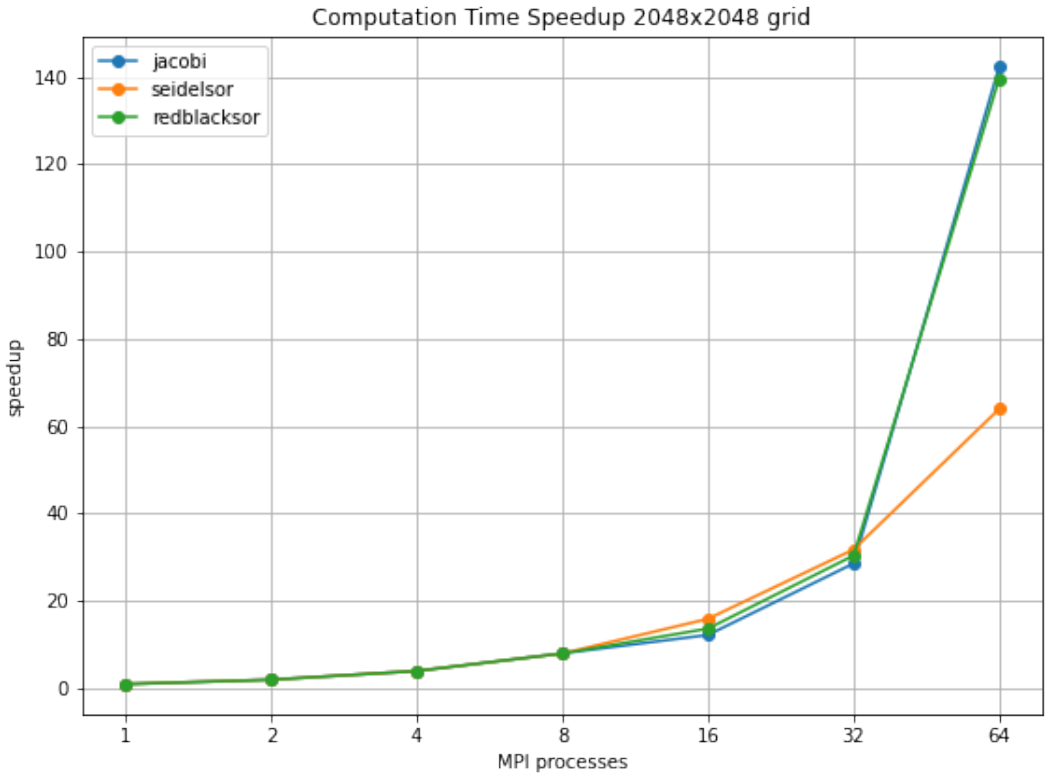
Φαίνεται ξεκάθαρα από τον πίνακα των αποτελεσμάτων πως η καλύτερη επιλογή είναι να χρησιμοποιήσουμε την μέθοδο RedBlackSOR.

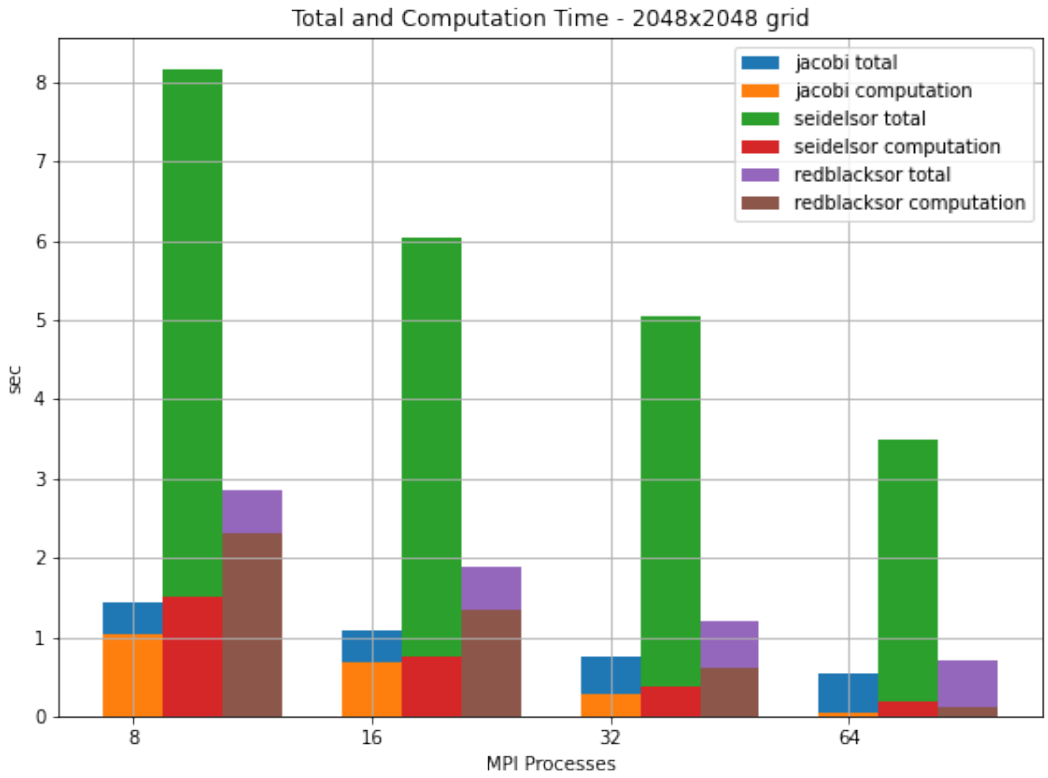
4 Μετρήσεις χωρίς έλεγχο σύγκλισης

Παρουσιάζονται τα αποτελέσματα των τριών μεθόδων για 256 Iterations και μεγέθη πίνακα 2048, 4096 και 6144 για 1, 2, 4, 8, 16, 32 και 64 MPI διεργασίες σε κάθε περίπτωση. Παρουσιάζονται τα διαγράμματα speedup και χρόνων υπολογισμού και εκτέλεσης. Έπειτα, ακολουθεί σχολιασμός αυτών.

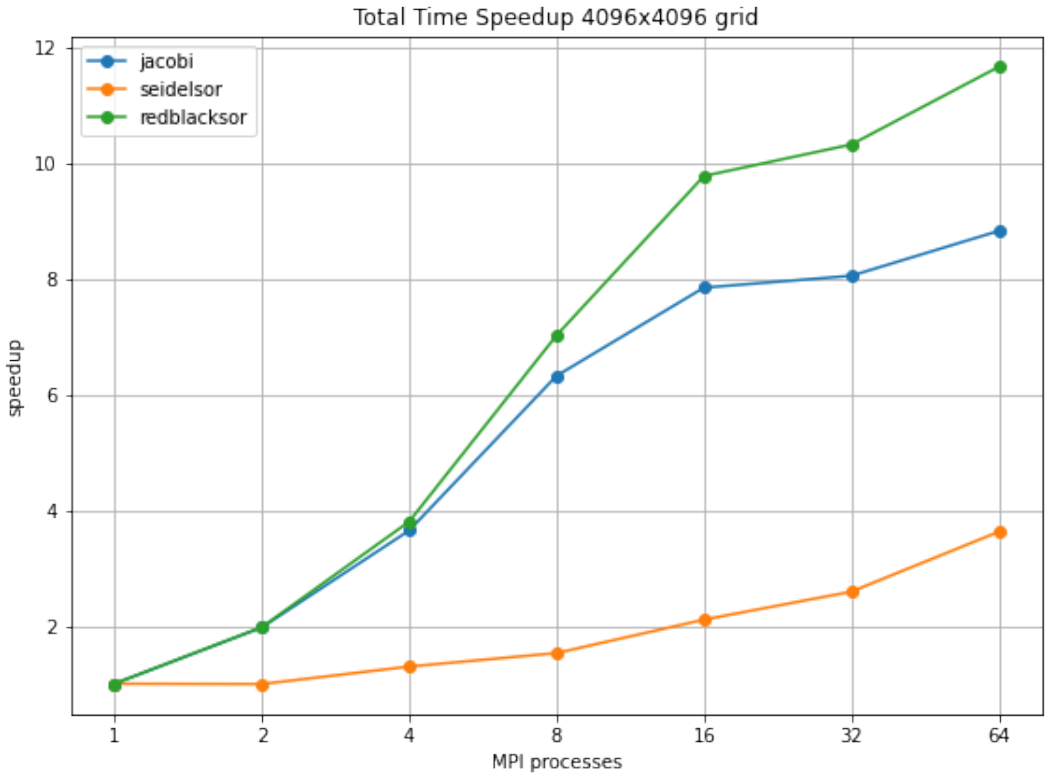
4.1 Grid 2048x2048

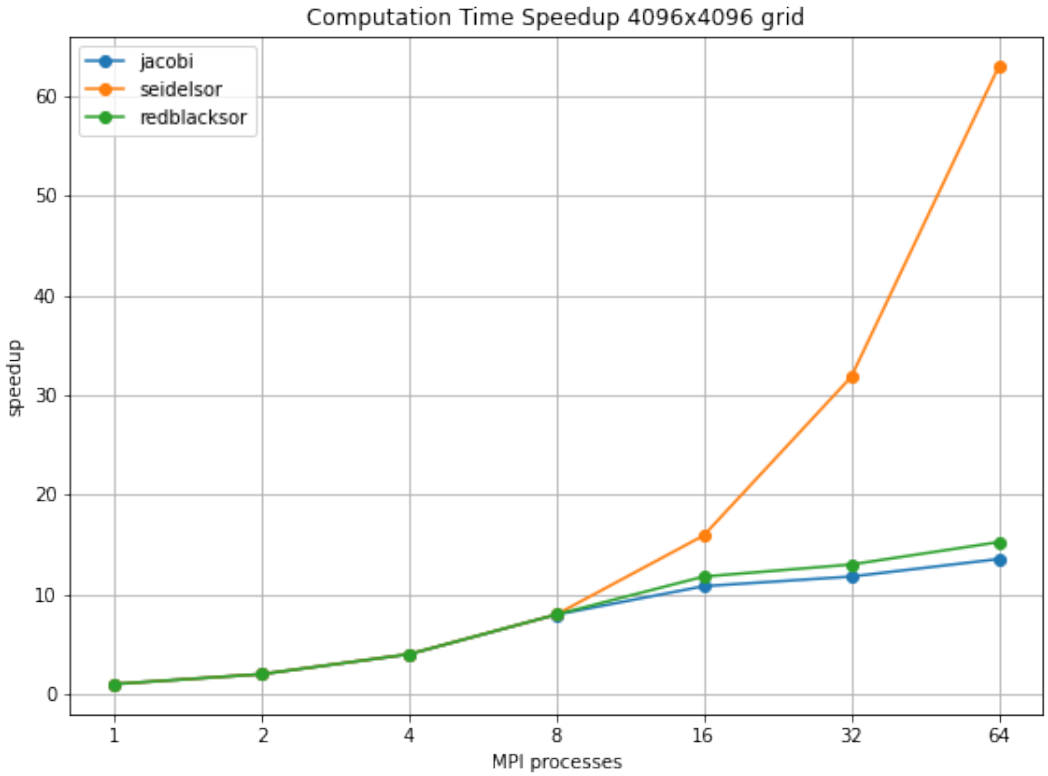


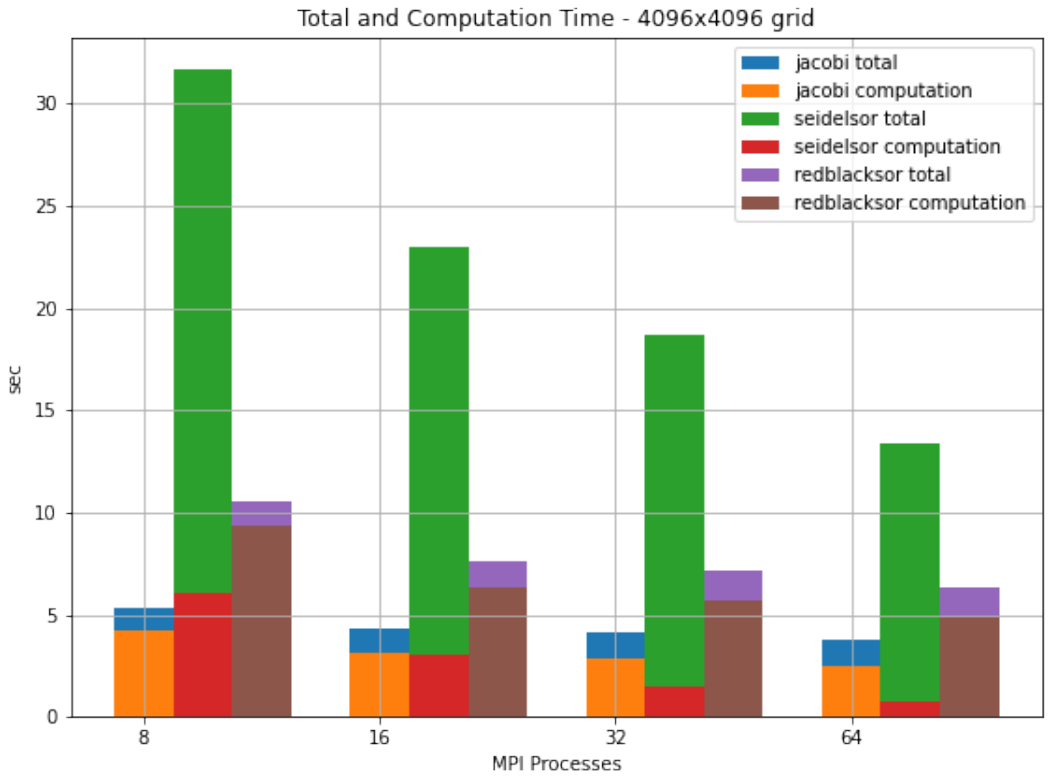



Total and Computation Time - 2048x2048 grid
jacobi total
jacobi computation
seidelsor total
seidelsor computation
redblacksor total
redblacksor computation
sec
0
1
2
3
4
5
6
7
8
8
16
32
64
MPI Processes

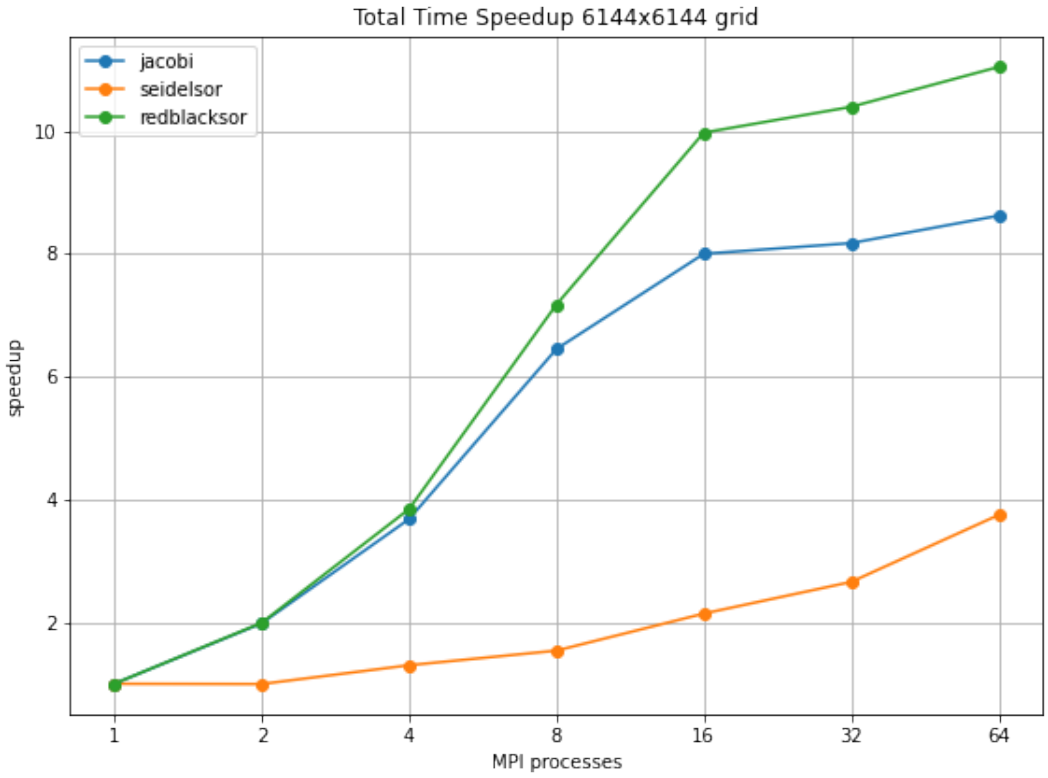
4.2 Grid 4096x4096

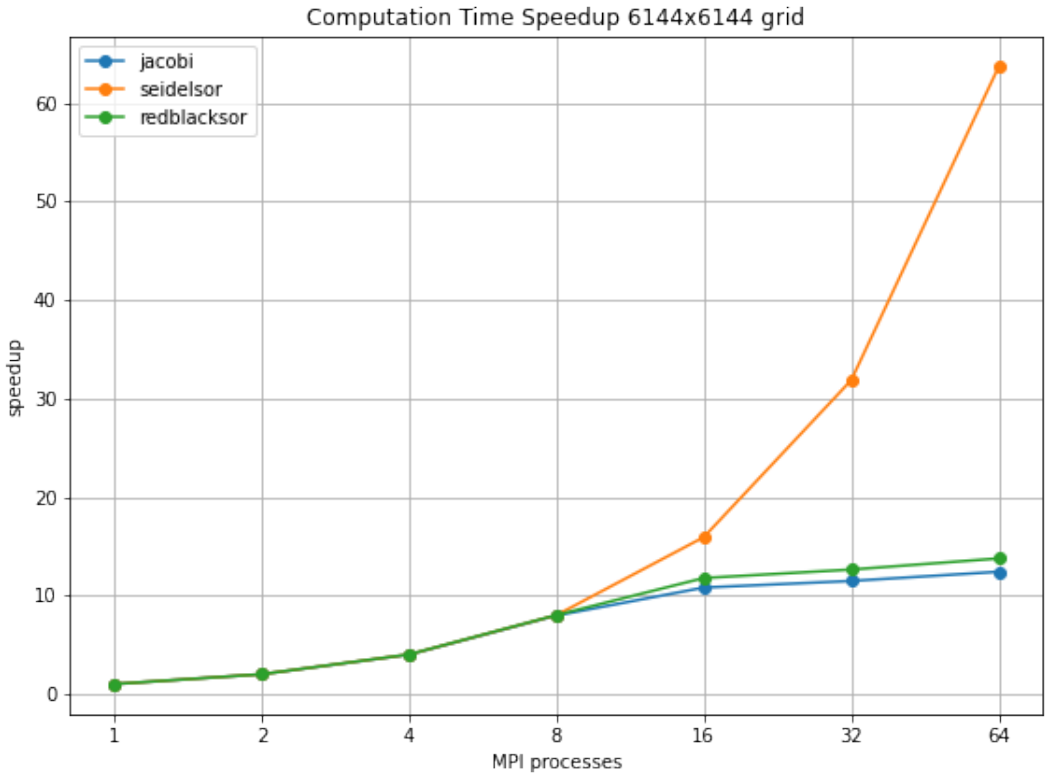


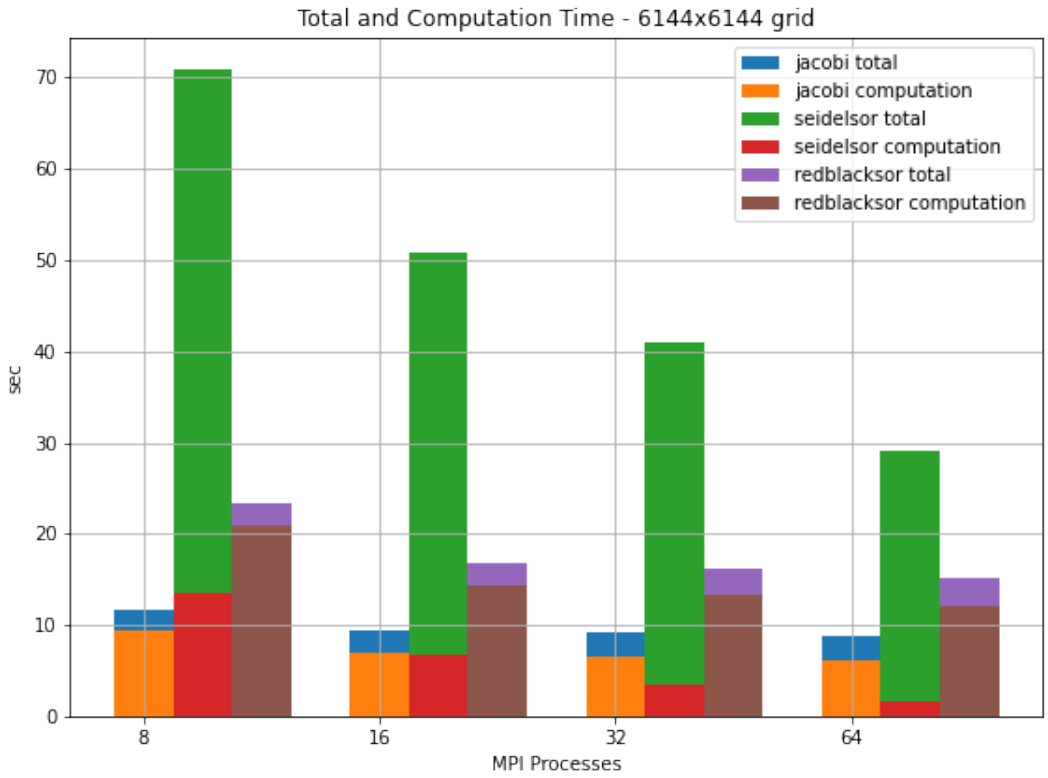



Total and Computation Time - 4096x4096 grid
jacobi total
jacobi computation
seidelsor total
seidelsor computation
redblacksor total
redblacksor computation
sec
0
5
10
15
20
25
30
8
16
32
64
MPI Processes

4.3 Grid 6144x6144







4.4 Παρατηρήσεις

Διαπιστοώνουμε τα ακόλουθα ως προς την κλιμακωσιμότητα των τριών υπολογιστικών μεθόδων:

- Από τα διαγράμματα χρόνου, για σταθερό αριθμό επαναλήψεων, ο Jacobi είναι ο ταχύτερος αλγόριθμος. Αυτό συμβαίνει γιατί έχει λιγότερο κόστος επικοινωνίας από τον Gauss-Seidel και λιγότερο υπολογιστικό κόστος από τον RedBlack.
- Καλύτερη κλιμάκωση πετυχαίνει ο Gauss-Seidel αλγόριθμος, εξαιρώντας το μέγεθος του πίνακα (η πιο κυρτή καμπύλη).
- Μεγαλύτερο total speedup πετυχαίνει ο RedBlackSOR. Ωστόσο, δεν πρέπει να παραβλέψουμε πως ο χρόνος υπολογισμού (computation time) καταλαμβάνει σχεδόν όλο τον χρόνο εκτέλεσης του αλγορίθμου. Από ένα σημείο και μεά, λόγω του υπολογιστικού κόστους, ο RedBlack-SOR δεν κλιμακώνει.
- Συνολικά, ο Gauss-Seidel SOR, κλιμακώνει καλύτερα και ταυτόχρονα μειώνει σταθερά τον συνολικό χρόνο εκτέλεσης (total time) παρά το κόστος επικοινωνίας που αντιμετωπίζει.
- Οι Jacobi και RedBlackSOR δεν κλιμακώνουν τόσο καλά διότι τα blocking SendRecv εμποδίζουν την επιπλέον μείωση του χρόνου επικοινωνίας.