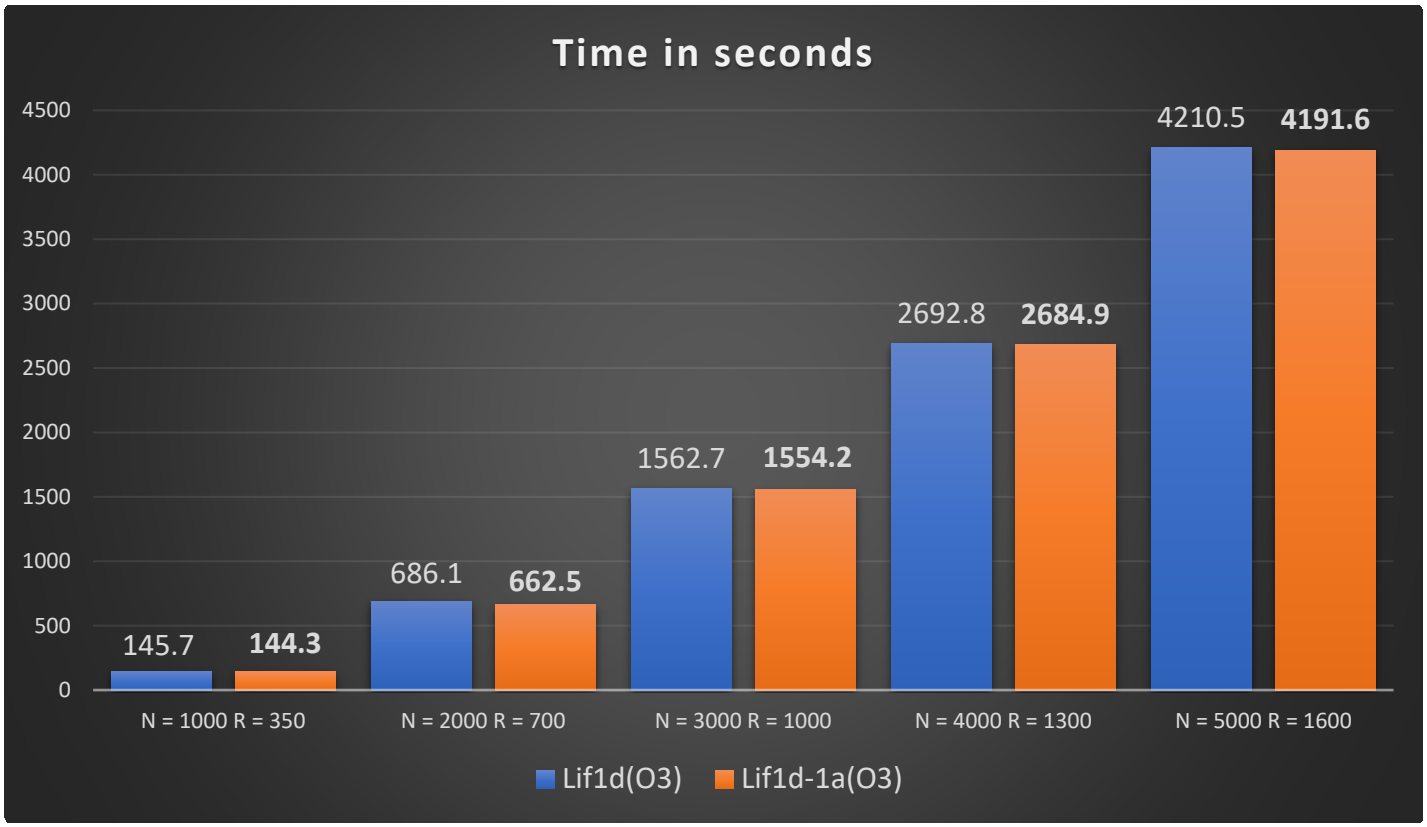
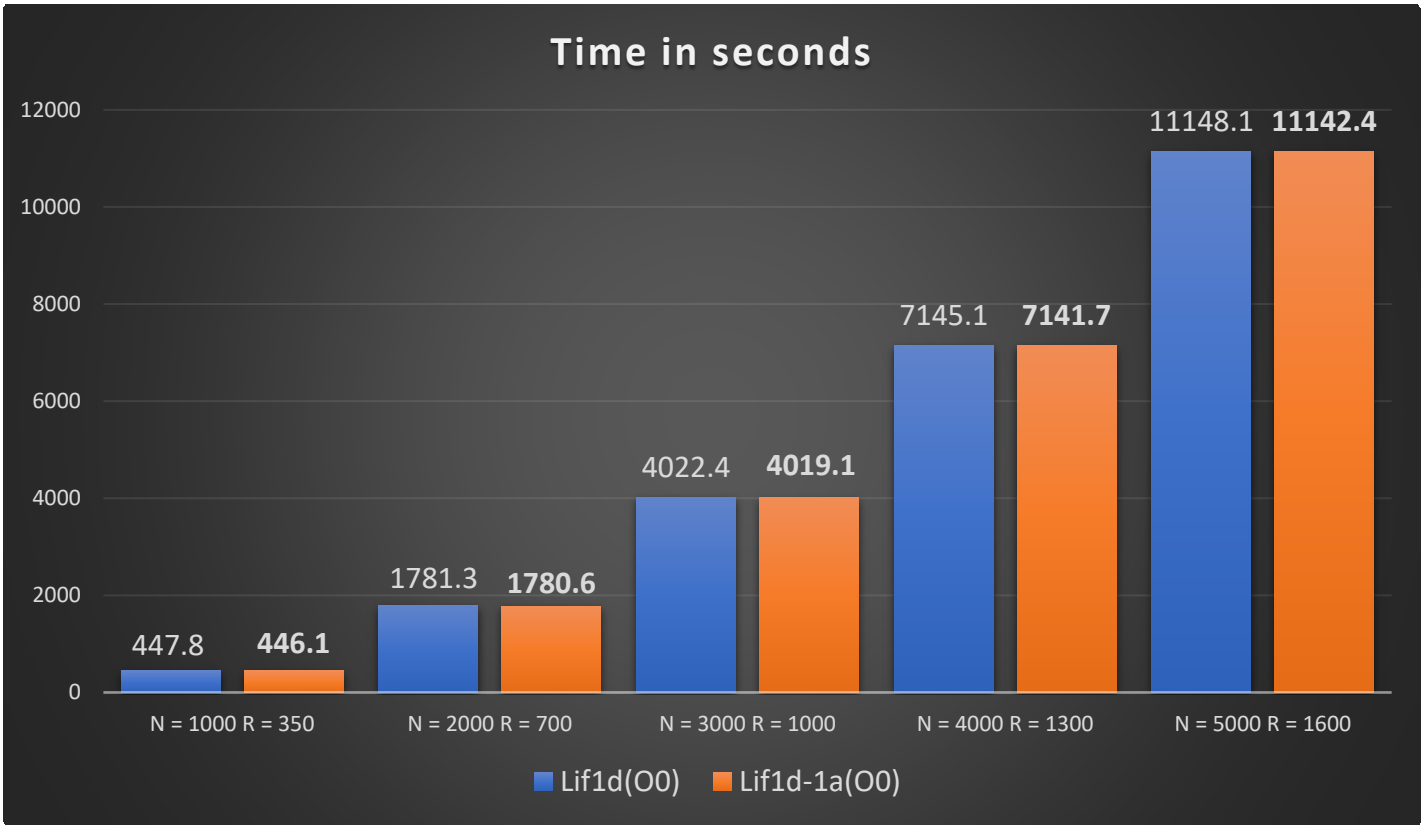
2018

# PROJECT ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

## ΑΝΑΦΟΡΑ- ΜΕΤΡΗΣΕΙΣ

ΚΑΤΣΟΣ ΠΑΝΑΓΙΩΤΗΣ – 1041772
ΛΙΟΥΠΗΣ ΠΑΝΑΓΙΩΤΗΣ – 1043795
ΠΑΠΑΔΟΠΟΥΛΟΣ ΠΑΝΤΕΛΗΣ – 1041854
ΠΥΡΓΑΣ ΑΘΑΝΑΣΙΟΣ – 1041866

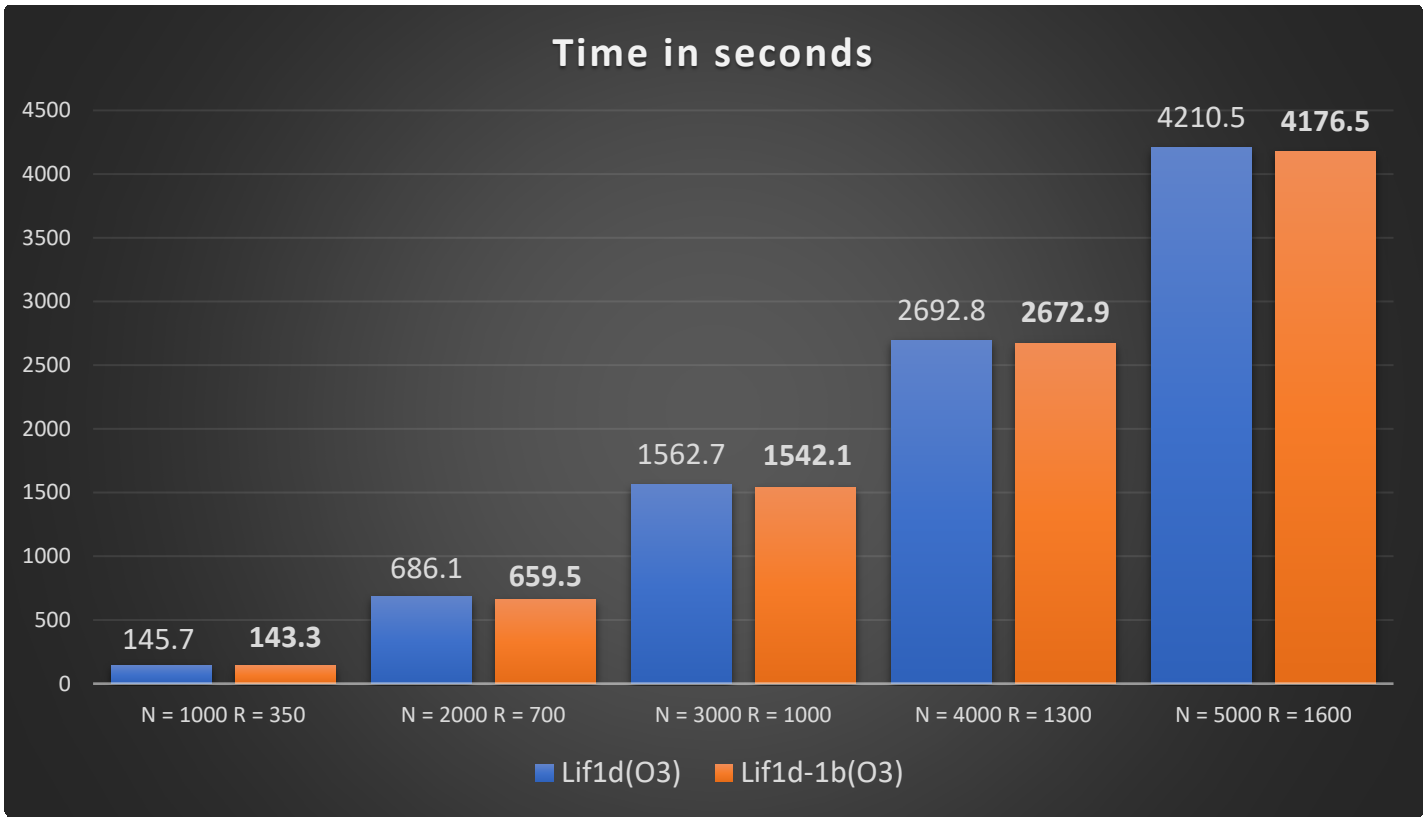


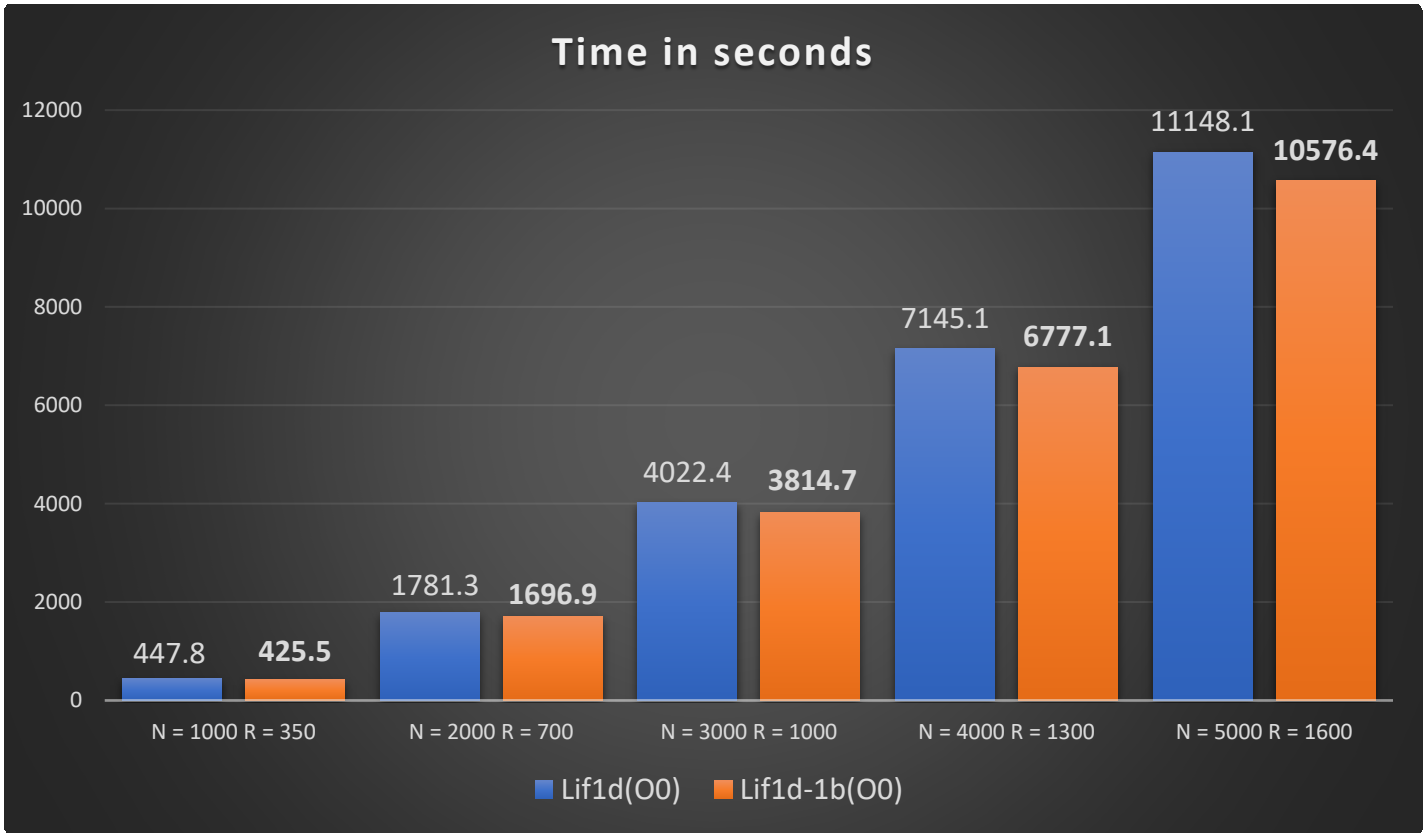




Lif1d-1a measurements

Μια πιο αποδοτική προσέγγιση για τη μείωση των αντιγραφών, που απαιτούνται να γίνουν στο διάνυσμα *u*, είναι η σύγκριση των νέων τιμών του διανύσματος *uplus* με την τιμή του ορίου του δυναμικού (threshold). Εάν η νέα τιμή υπερβαίνει την τιμή του ορίου, τότε αυτή μηδενίζεται για τον συγκεκριμένο νευρώνα, αλλιώς αντιγράφεται απευθείας στο διάνυσμα *u*. Έτσι αποφεύγουμε έναν αισθητό αριθμό αντιγραφών. Οι βελτιώσεις στο χρόνο φαίνονται στα 2 παραπάνω διαγράμματα (Lif1d-1a).

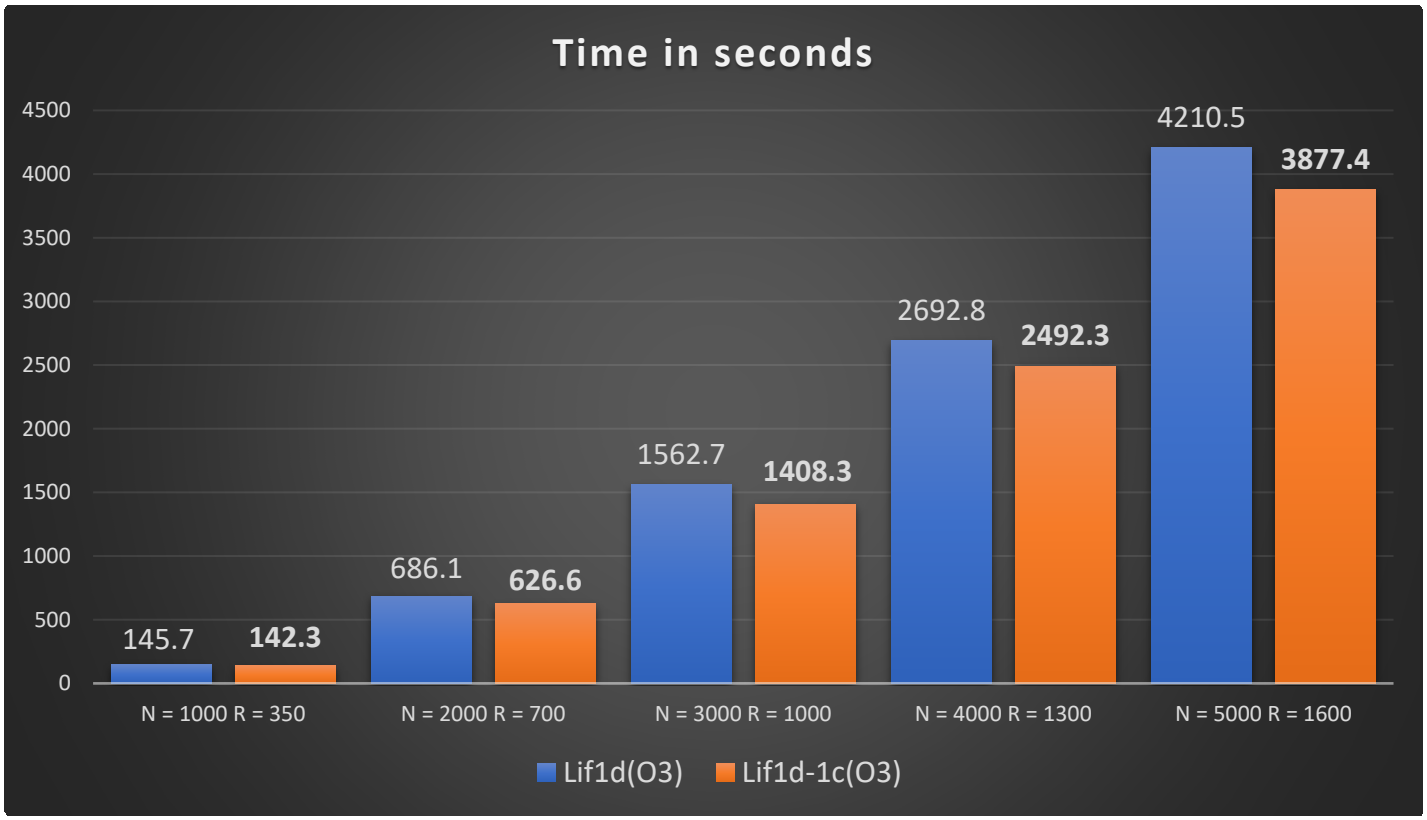
Comments on Lif1d-1a

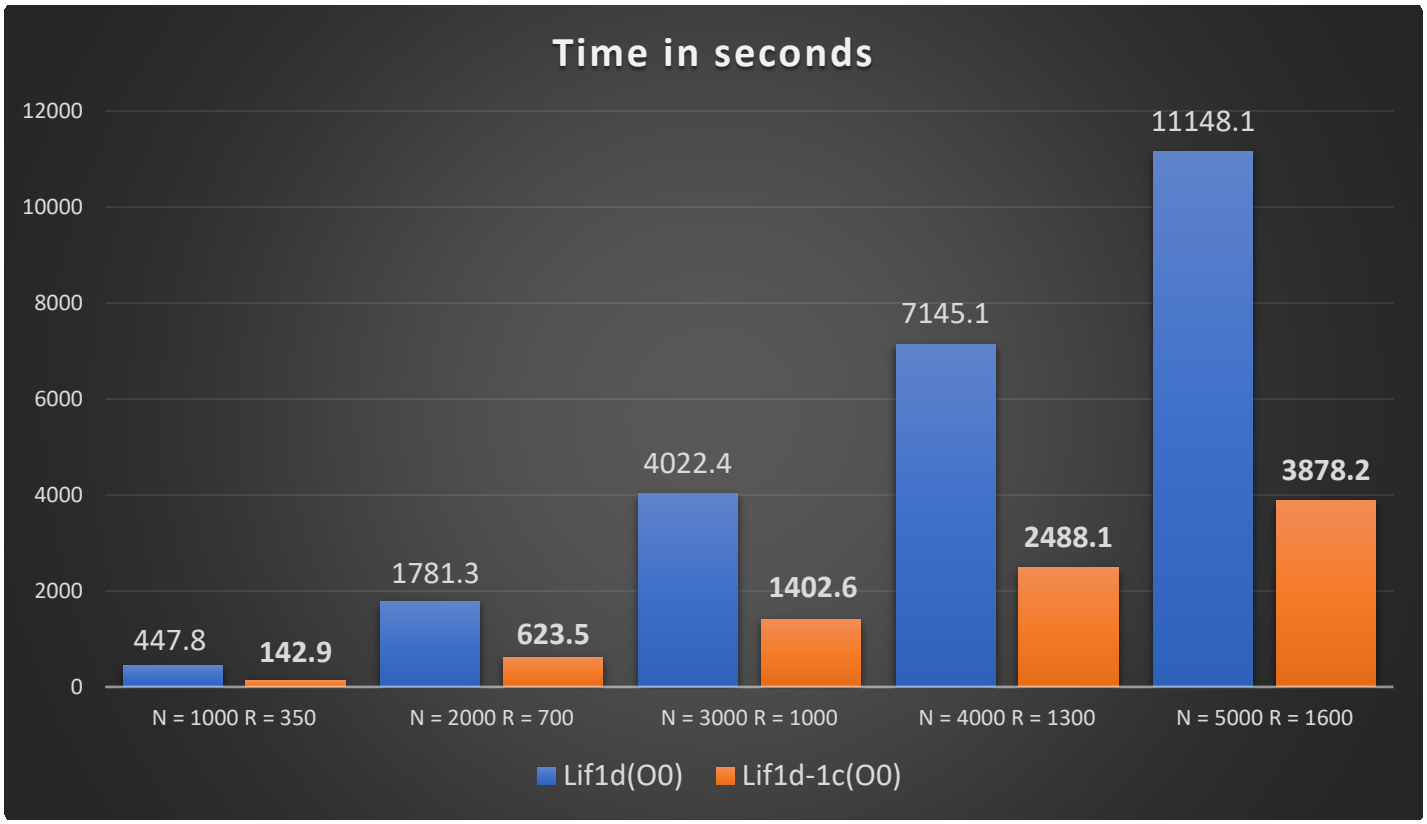



Lif1d-1b measurements

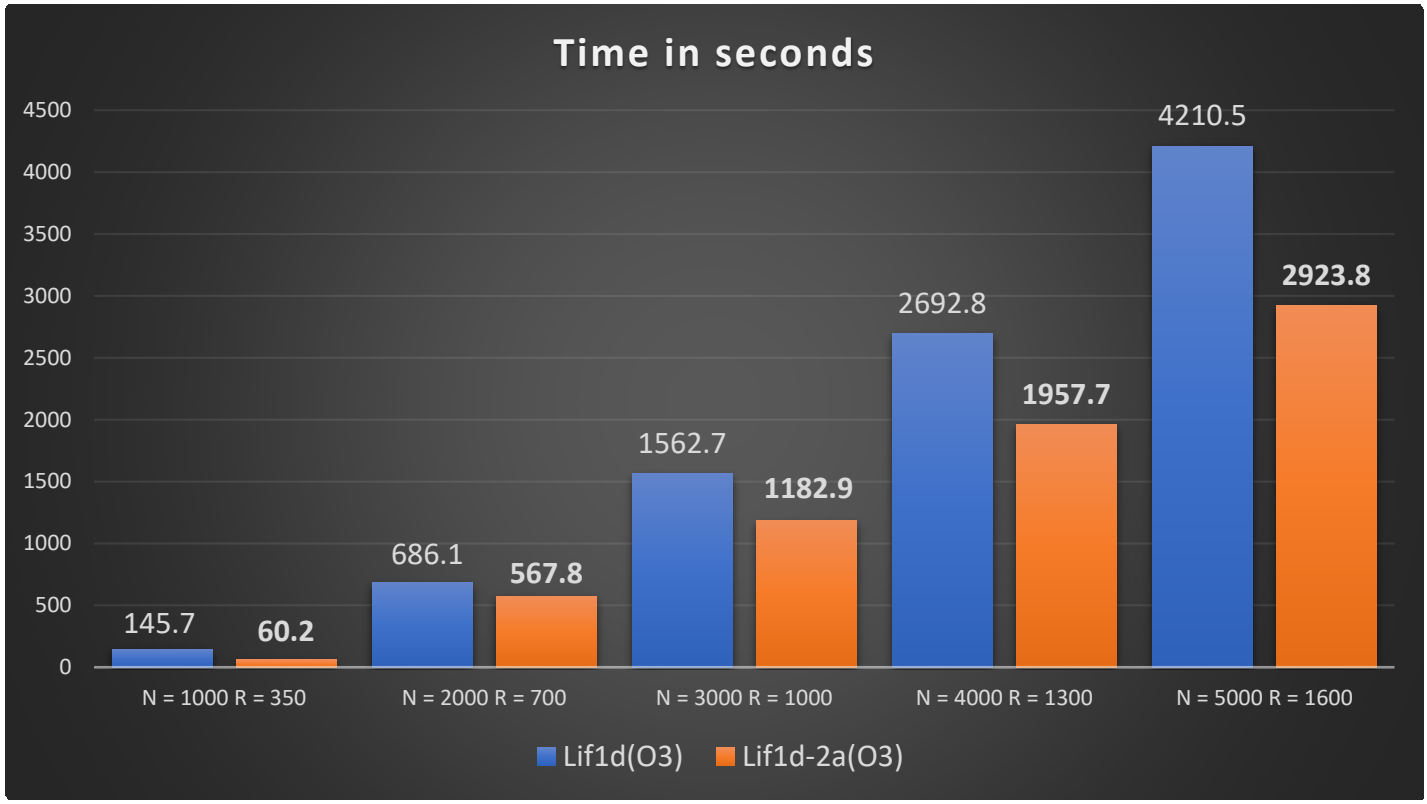
Αρχικά, με σκοπό την αναδιοργάνωση των υπολογισμών που χρειάζονται σε κάθε επανάληψη, εφαρμόστηκε η επιμεριστική ιδιότητα στην πράξη sigma * (u[j] – u[i]). Έτσι, για το κομμάτι του sigma * u[i], υπολογίζονται σε ένα διάνυσμα sum εκτός του βρόχου “it” τα επιμέρους αθροίσματα του μητρώου sigma μια φορά κατά την εκτέλεση του αλγορίθμου, λόγω του ότι το μητρώο αυτό παραμένει σταθερό. Ύστερα, αυτά πολλαπλασιάζονται με το διάνυσμα u εκτός του βρόχου “j” και αποθηκεύονται σε μια προσωρινή μεταβλητή sum2. Για την πράξη sigma * u[j] διατηρούνται οι υπολογισμοί μέσα στο βρόχο “j” από τον οποίο εξερτάται. Τέλος, γίνονται οι πράξεις για το διάνυσμα *uplus* με τη χρήση των μεταβλητών που δημιουργήθηκαν. Οι βελτιώσεις στο χρόνο φαίνονται στα 2 παραπάνω διαγράμματα (Lif1d-1b).

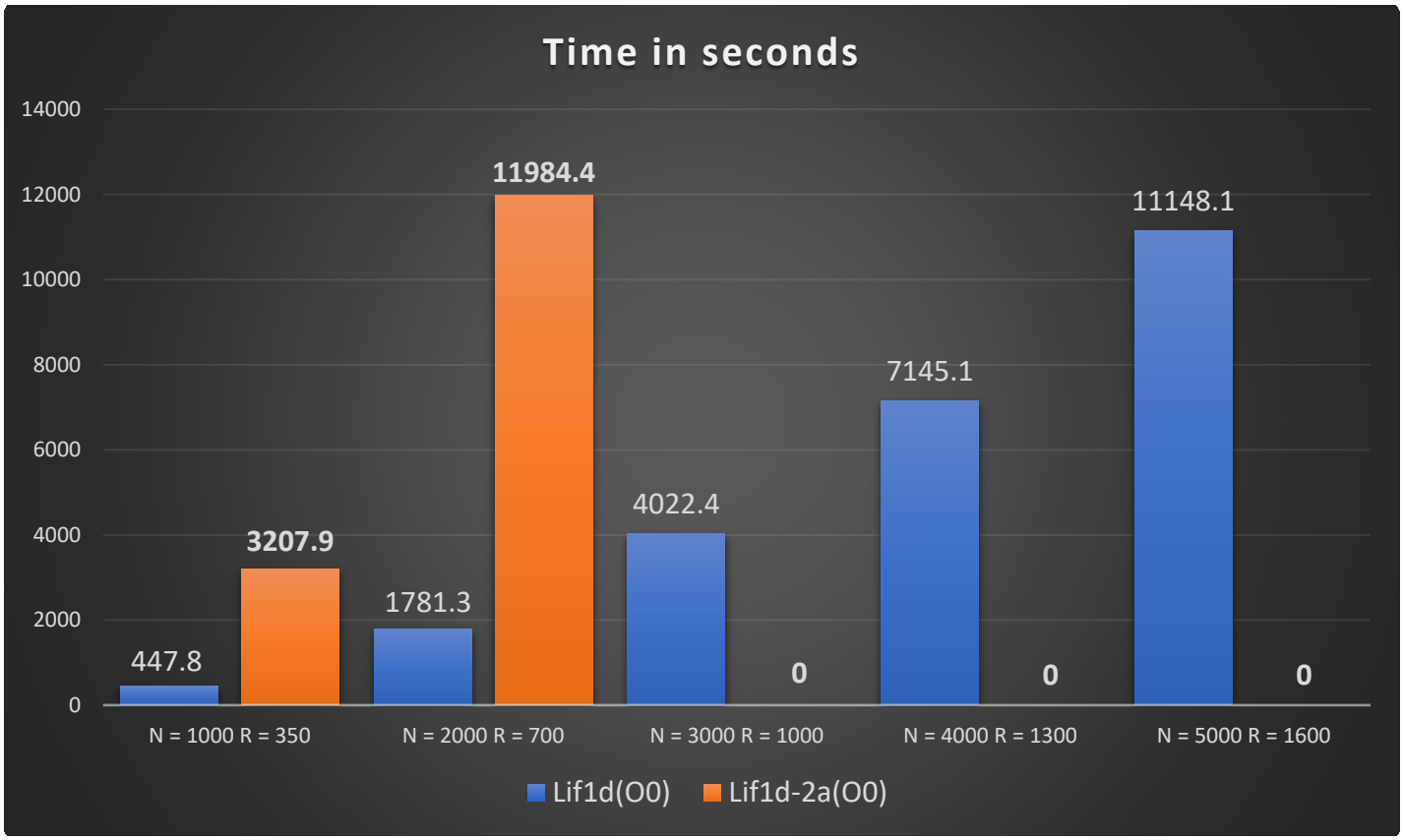



Lif1d-1c measurements

Παρατηρούμε πως στο βρόχο “j” στη μεταβλητή *sum1* αποθηκεύεται το εσωτερικό γινόμενο κάθε γραμμής του μητρώου *sigma* με το διάνυσμα *u.* Ο υπολογισμός αυτός μπορεί εύκολα να αντικατασταθεί με τη συνάρτηση *ddot* της βιβλιοθήκης *blas* με την κατάλληλη μετατροπή στον *pointer* του μητρώου *sigma,* ο οποίος μεταβάλλεται κατάλληλα με βάση το βήμα του βρόχου “i”. Οι βελτιώσεις στο χρόνο φαίνονται στα 2 παραπάνω διαγράμματα (Lif1d-1c).



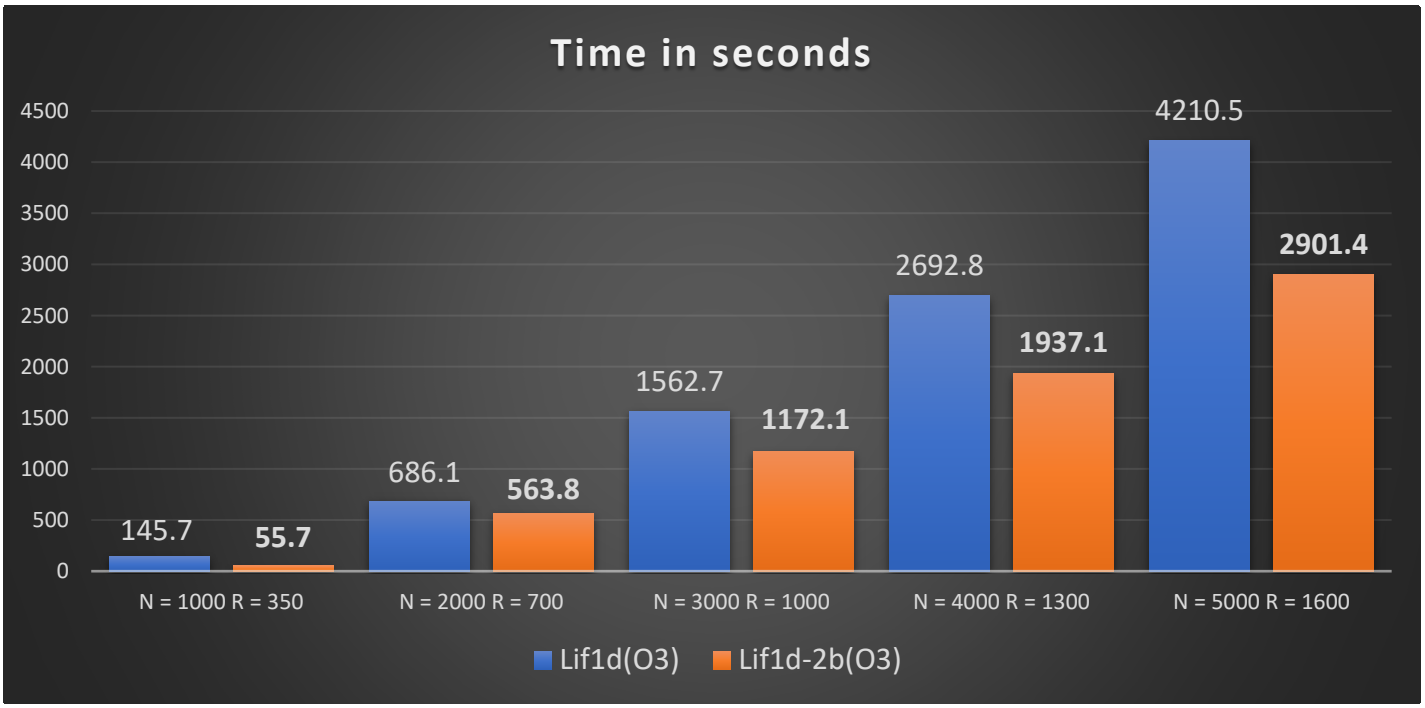



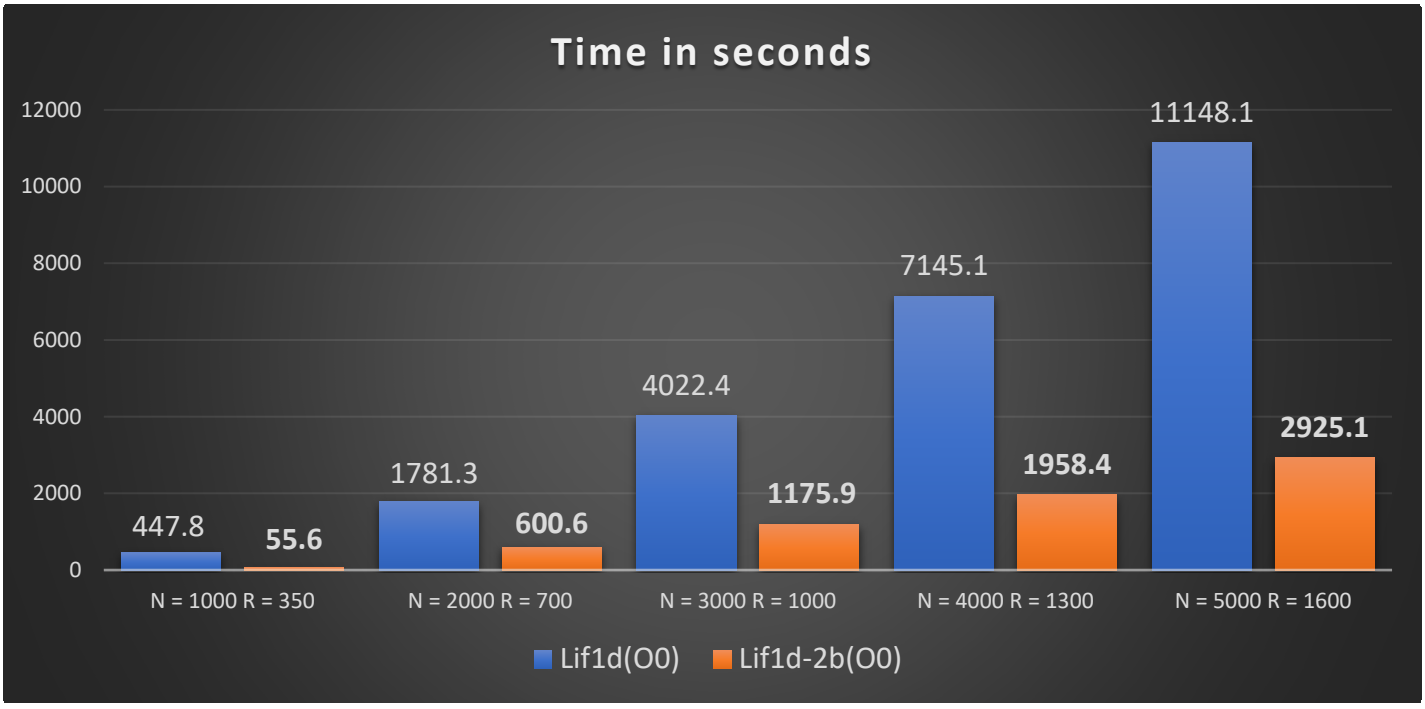

Lif1d-2a measurements

Η παραλληλοποίηση του βρόχου “it” ήταν αδύνατη, διότι προέκυψαν εξαρτήσεις δεδομένων. Πιο συγκεκριμένα τα βήματα του “it” διαμοιράζονται στα νήματα, με αποτέλεσμα οι τιμές στα διανύσματα *u, uplus* παρόλο που γράφονται με σωστή σειρά, να ανακατεύονται κατά την εγγραφή τους στα τελικά αρχεία. Αυτό γίνεται γιατί υπάρχει η πιθανότητα ενα νήμα που έχει την τιμή 100000 του “it” να ολοκληρωθεί πιο γρήγορα από ένα που έχει την τιμή 20000. Επομένως, χρειάστηκε η χρήση της οδηγίας *ordered* για να αποφευχθεί το παραπάνω πρόβλημα, οδηγόντας στην ψευδο-παραλληλοποίηση του αλγορίθμου χωρίς να υπάρχει κάποιο κέρδος σε χρόνο.

Η παραλληλοποίηση του βρόχου “i” ήταν η πιο κερδοφόρα σε χρόνο. Στην περίπτωση αυτή επιτυγχάνεται τόσο η σωστή σείρα υπολογισμού των διανυσμάτων *u, plus,* όσο και η σωστή (σειριακή) αποθήκευση των αποτελεσμάτων στα αρχεία εξόδου. Κάθε νήμα αναλαμβάνει ένα δικό του αριθμό από τις τιμές της επανάληψης. Στα παραπάνω διανύσματα κατά την εγγραφή των τιμών τους, δεν έχει σημασία η σείρα με την οποία θα γίνουν π.χ αν τελειώσει πρώτα το νήμα που έχει αναλάβει την τιμή 100 ή το νήμα με την τιμή 50. Οι βελτιώσεις στο χρόνο φαίνονται στα 2 παραπάνω διαγράμματα (Lif1d-2a).

Η παραλληλοποίηση του βρόχου “j” δεν είχε κάποια ιδιαίτερη βελτίωση στο χρόνο εκτέλεσης του αλγορίθμου αφού αποτελούσε ενα πολύ μικρό κομμάτι των πράξεων που απαιτούνται. Η παραλληλοποίηση επιτεύχθηκε με τη χρήση της οδηγίας reduction(+:sum1), για την μεταβλητή sum1.




Lif1d-2b measurements

Η καλύτερη, χρονικά, υλοποίηση ήταν η παραλλοποίηση του βρόχου “i”. Αντικαθηστώντας τον βρόχο “j” με τη συνάρτηση ddot της βιβλιοθήκης blas, προέκυψε πρόβλημα με τον υπολογισμό των αποτελεσμάτων λόγω του *pointer* της *ddot* στο μητρώο *sigma,* ο οποίος έβγαινε εκτός περιοχής μνήμης (segmentation fault). Με τη χρήση της οδηγίας *private* τόσο στο μετρητή *i*, όσο και στις προσωρινές μεταβλητές αποθήκευσης των αποτελεσμάτων *sum2*, *sumdot*, το κάθε νήμα έχει ένα δικό του αντίγραφο της εκάστοτε μεταβλητής, λύνοντας το προαναφερθέν πρόβλημα. Οι βελτιώσεις στο χρόνο φαίνονται στα 2 παραπάνω διαγράμματα (Lif1d-2b).

Παρατηρήθηκε πως στις 2 πρώτες περιπτώσεις (1a, 1b) οι χρονικές βελτιώσεις, είτε με βελτιστοποιήσεις του μεταφραστή είτε χωρίς, ήταν της ίδιας τάξης, αυτό είναι λογικό αφού ο αλγόριθμος βασίζεται στη γλώσσα *c* χωρίς να χρησιμοποιούνται εξωτερικές βιβλιοθήκες (*blas, openmp*). Τη στιγμή που εισήχθηκε η βιβλιοθήκη *blas* φαίνεται πως κατάφερε να ανταπεξέλθει τον μη-βεπτιστοποιήμενο μεταφραστή, κρατώντας τον χρόνο εκτέλεσης κοντά σε σχέση με τον βελτιστοποιημένο. Ακριβώς το ίδιο φαινόμενο παρατηρήθηκε και στην παραλληλοποίηση του αλγορίθμου. Χωρίς τη βιβλιοθήκη *blas* και τις βελτιστοποιήσεις ο αλγόριθμος χρειάστηκε περίπου 3 ώρες για τους υπολογισμούς με 2000 νευρώνες ενώ για 3000 πάνω από 7 (εκτίμηση). Εισάγωντας πάλι τη βιβλιοθήκη blas και των private μεταβλητών παρατηρείται ότι ο χρόνος επανέρχεται στο επίπεδο του βελτιστοποιημένου μεταφραστή.

Comments on compiler optimization