Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Συστήματα Παράλληλης Επεξεργασίας

2η Εργαστηριακή Άσκηση - Τελική Αναφορά

9ο Εξάμηνο - Ακαδημαϊκό Έτος 2020-2021

Γιαννόπουλος Εμμανουήλ - 03117031

Παγώνης Γεώργιος - 03117030

11 Δεκεμβρίου 2021

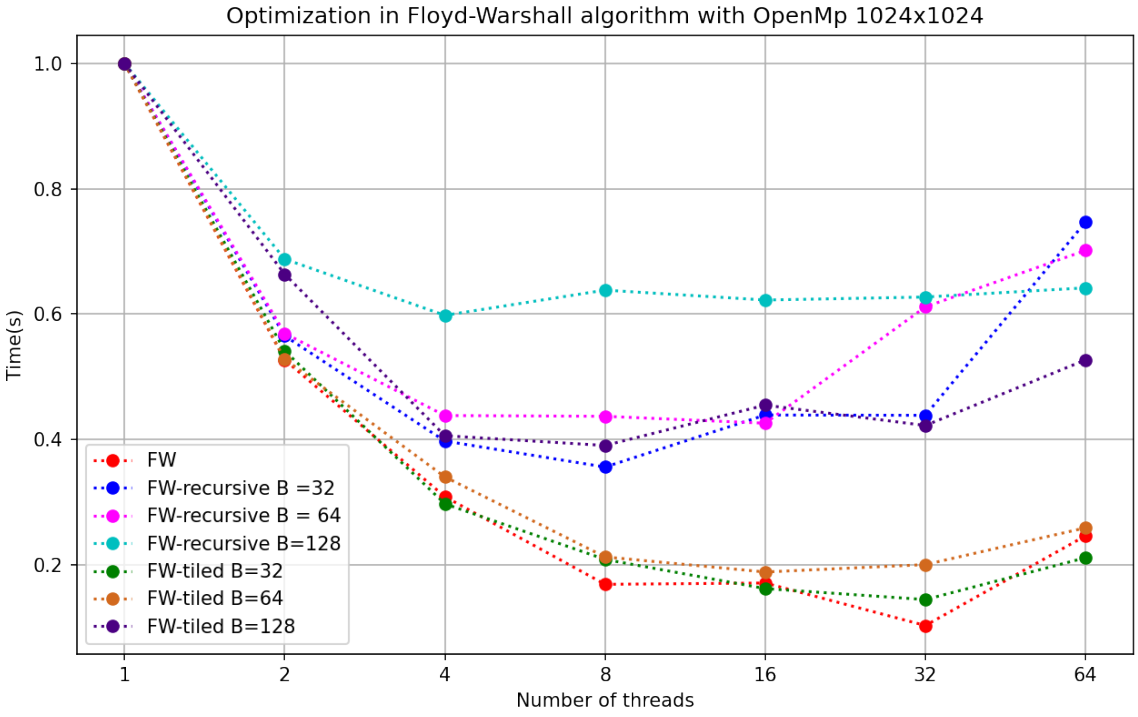
1 Εισαγωγή

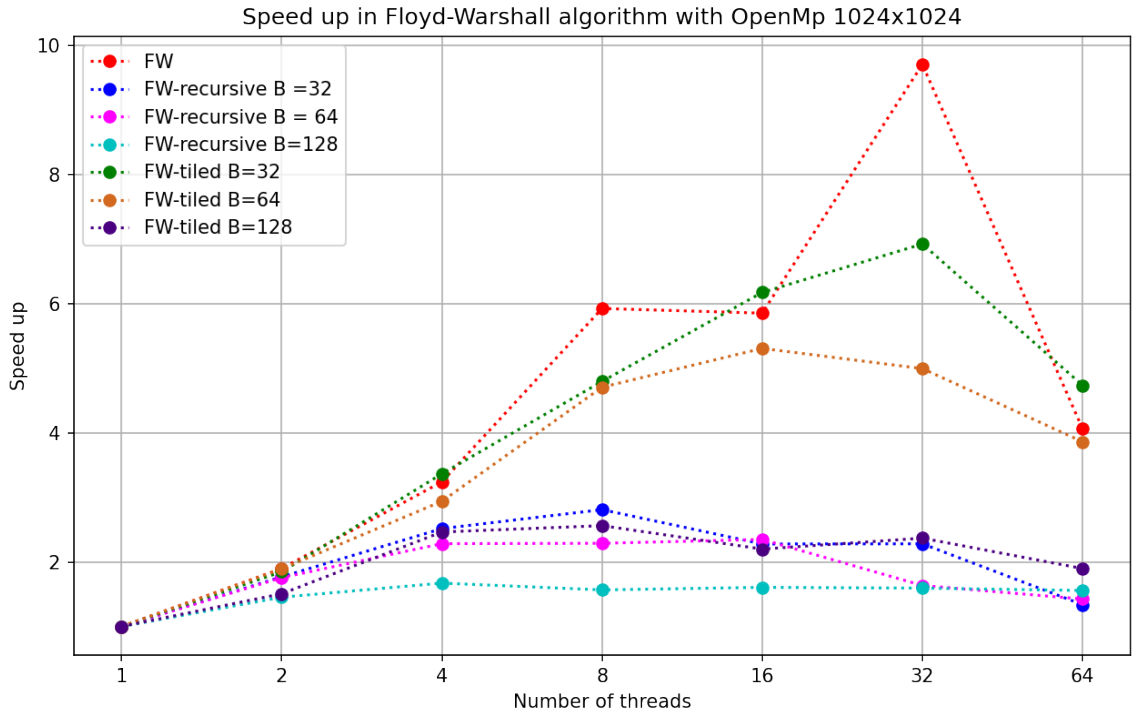
Σε αυτή την άσκηση σκοπός είναι να παραλληλοποιήσουμε τον αλγόριθμο εύρεσης συντομότερων μονοπατιών μεταξύ κάθε ζεύγους κόμβων **Floyd-Warshall**. Για το σκοπό αυτό θα μελετήσουμε 3 διαφορετικές υλοποιήσεις του αλγόριθμου (FW, FW-Recursive και FW-Tiled) και θα εξετάσουμε την παραλληλοποίηση που προσφέρει κάθε υλοποίηση καθώς και την επίδοσή τους.

2 OpenMP

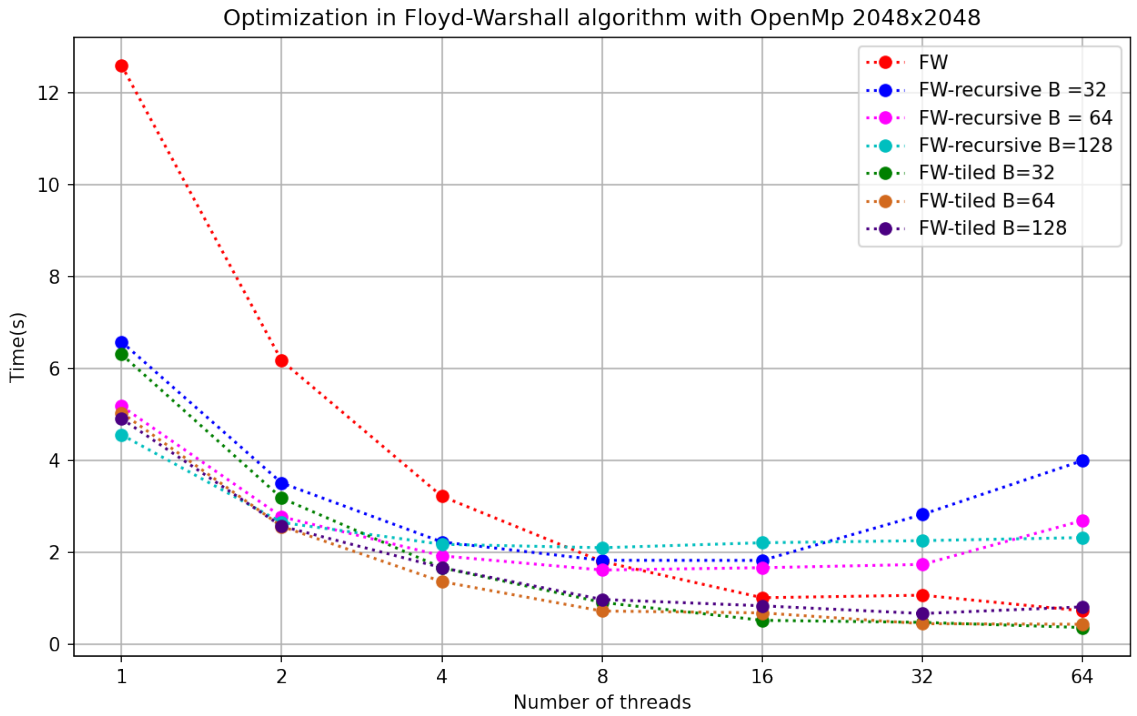
Αρχικά θα εξετάσουμε τις υλοποιήσεις μας σε OpenMP. Υλοποιήσαμε τις εξής εκδοχές: FW με parallel for, task-based FW-Recursive και FW-tiled με parallel for.

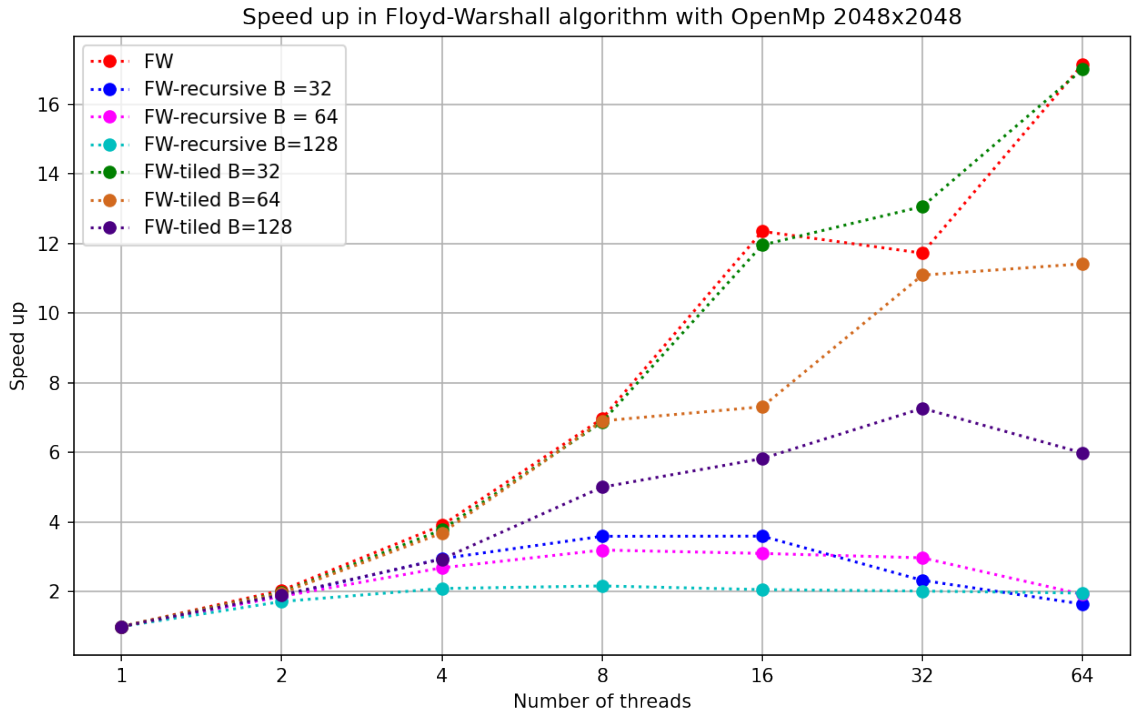
2.1 Size 1024x1024



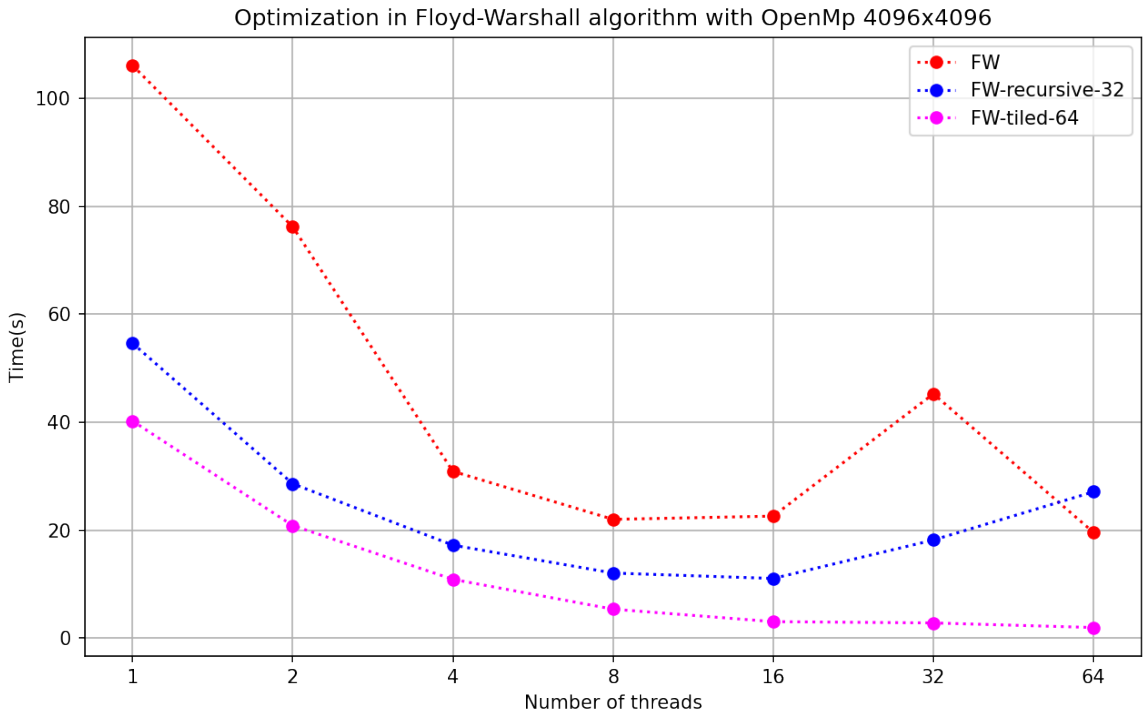


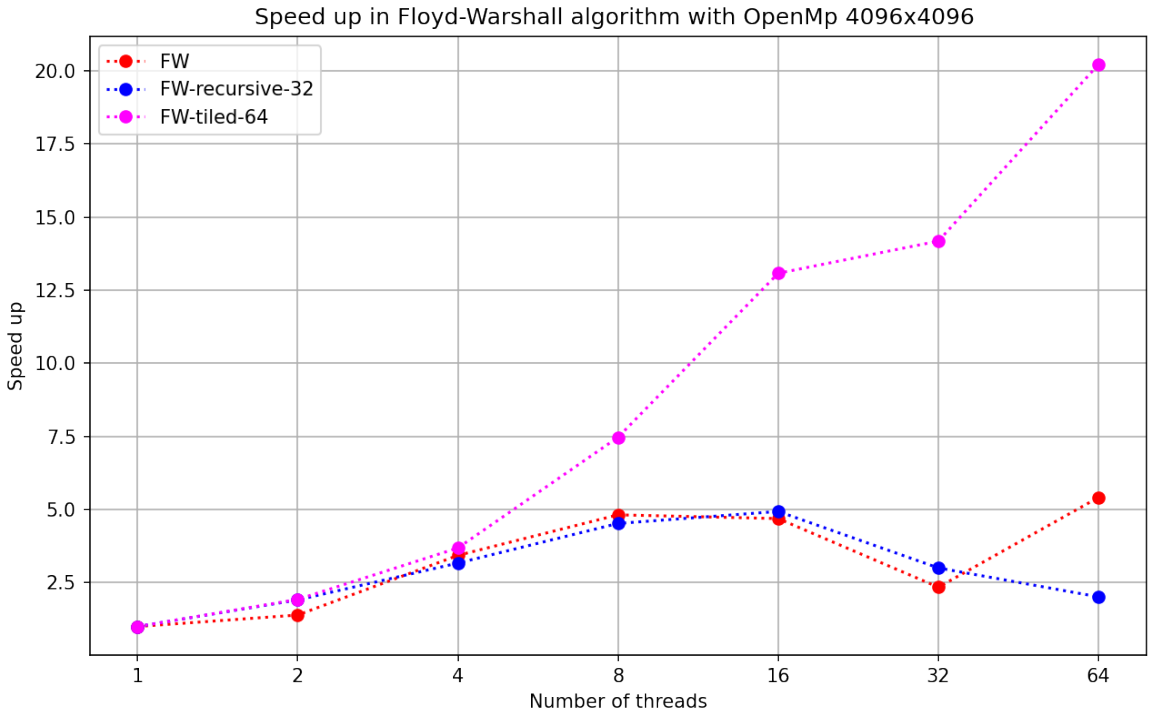
2.2 Size 2048x2048

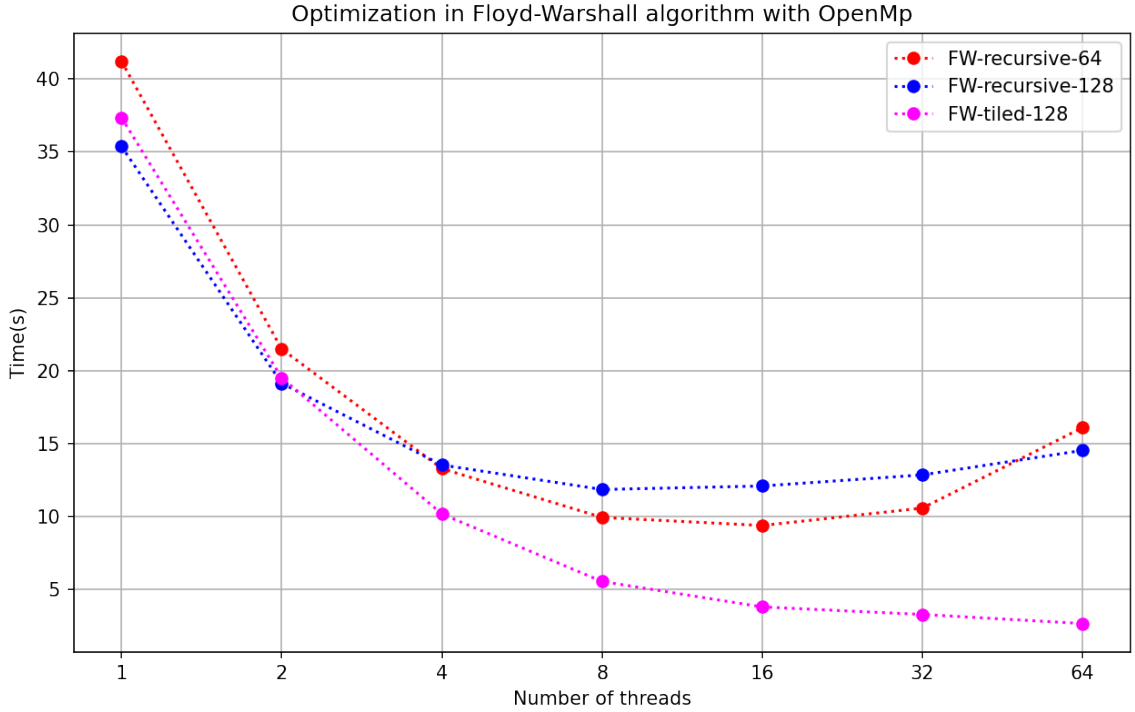


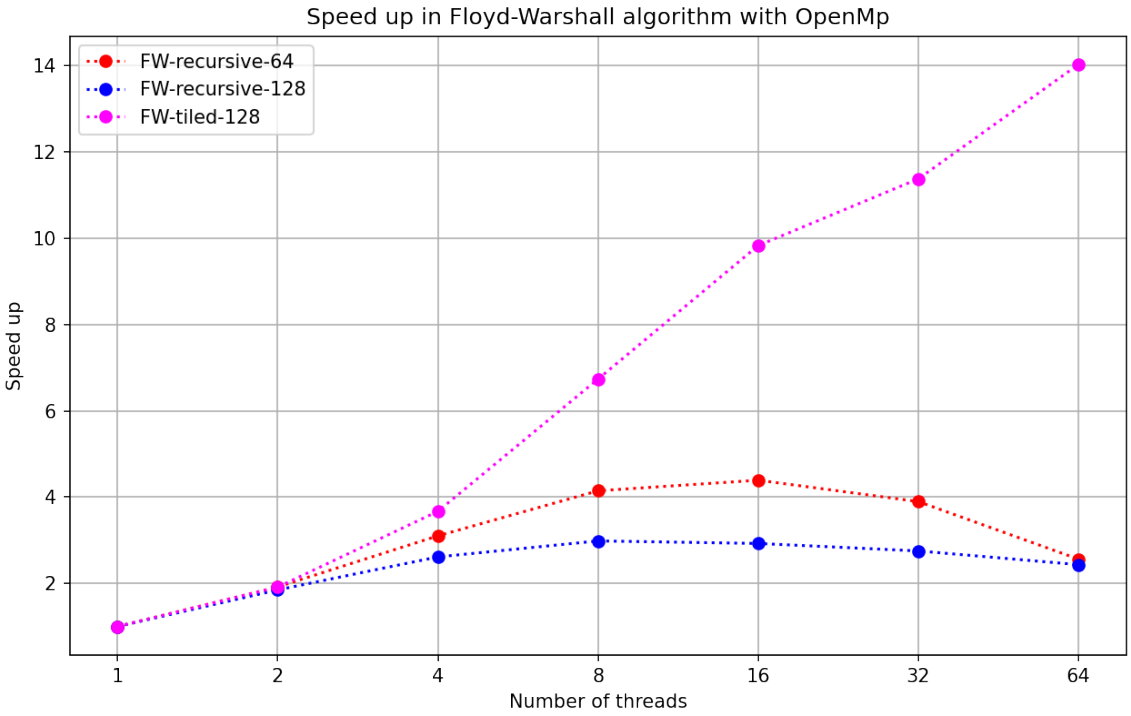


2.3 Size 4096x4096




Speed up in Floyd-Warshall algorithm with OpenMp 4096x4096
FW
FW-recursive-32
FW-tiled-64
Speed up
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
2
4
8
16
32
64
Number of threads


Optimization in Floyd-Warshall algorithm with OpenMp
FW-recursive-64
FW-recursive-128
FW-tiled-128
Time(s)
40
35
30
25
20
15
10
5
1
2
4
8
16
32
64
Number of threads



Παρατηρήσεις (κυρίως για size 4096x4096):

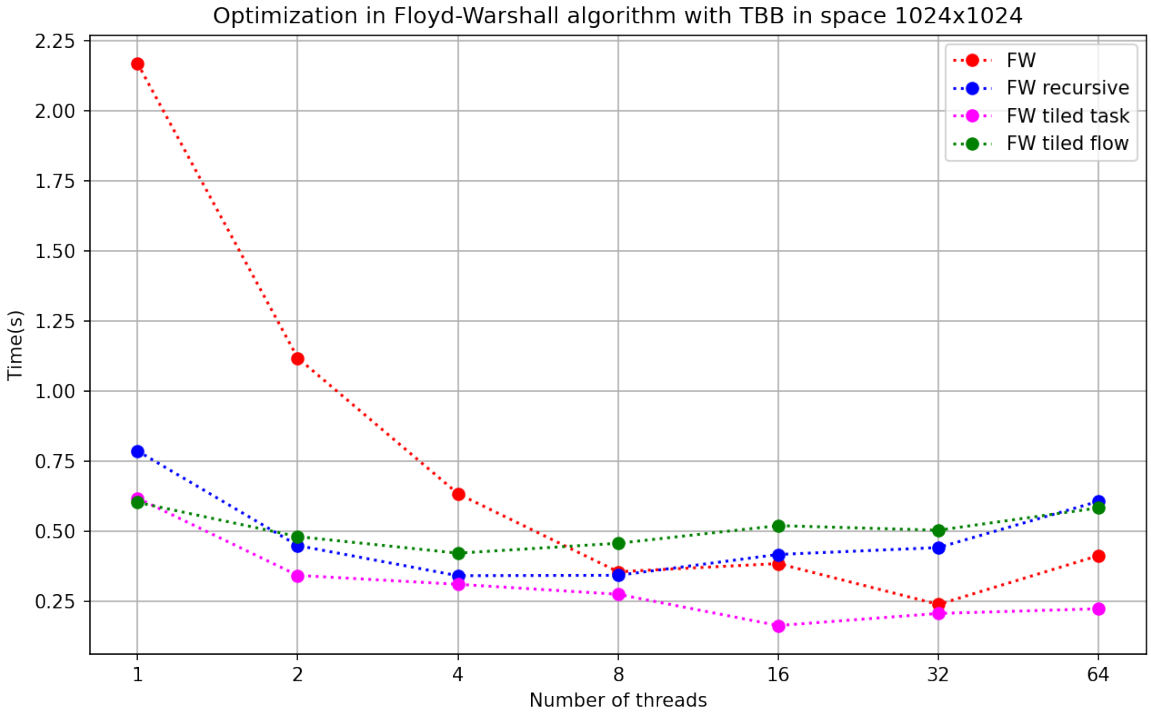
- FW: Παρατηρούμε ότι ο απλός FW αλγόριθμος δεν κλιμακώνει καλά. Δοκιμάσαμε παράλληλη αρχικοποίηση για να εκμεταλλευτούμε την first touch πολιτική του Linux αλλά το μέγιστο speedup ήταν λίγο πάνω απο 5 και το scalability φαίνεται να "παγώνει" από τα 8 threads. Υποθέτουμε ότι λόγω της αρχιτεκτονικής του sandman, όταν χρησιμοποιούνται παραπάνω από 1 nodes η επικοινωνία αρχίζει να κοστίζει παραπάνω από το όφελος της παραλληλοποίησης με περισσότερα νήματα.

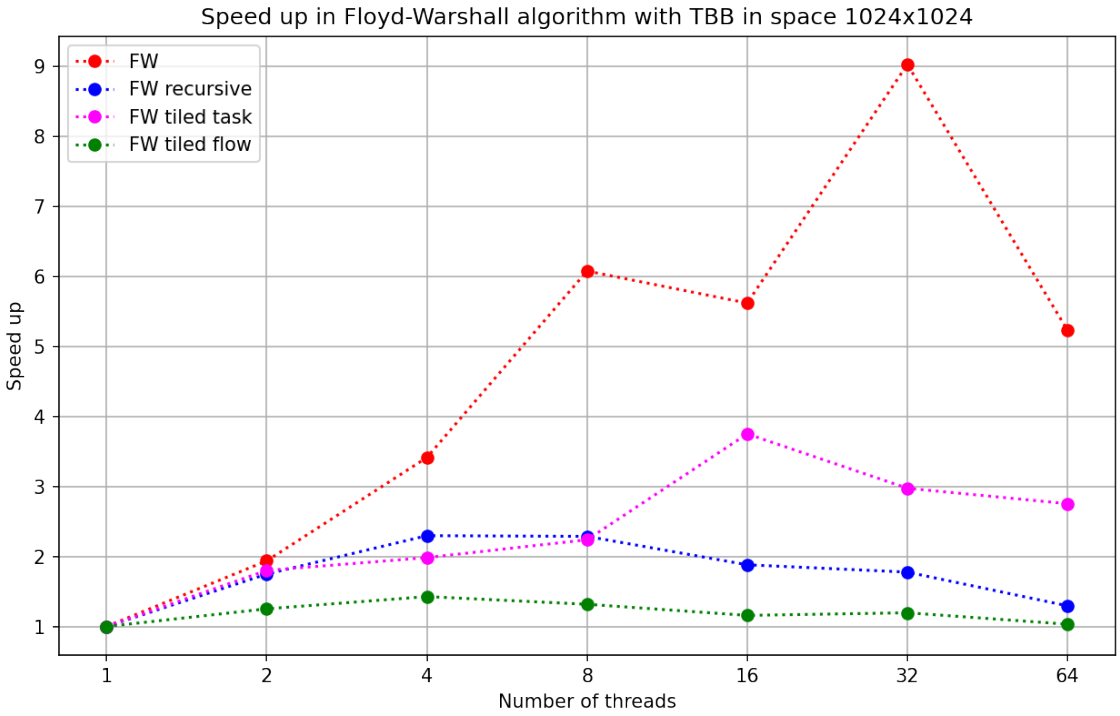
- FW-Recursive: Υλοποιήσαμε σε OpenMP τον αλγόριθμο FW-Recursive χρησιμοποιώντας τα tasks του OpenMP. Παρατηρούμε ότι το speedup κάνει peak στα 16 threads και μετά πέφτει. Με βάση τον τρόπο που δουλεύει το work stealing υποθέτουμε ότι δεν υπάρχει αρκετή "δουλειά" για να "κλέψουν" τα περισσότερα threads ώστε να δικαιολογεί τον περισσότερο χρόνο που χρειάζεται ο συγχρονισμός τους (λόγω και της αρχιτεκτονικής του sandman). Αξίζει να σημειωθεί ότι το speedup είναι μικρότερο από του απλού FW όμως μικρότεροι είναι και οι χρόνοι εκτέλεσης. Αυτό συμβαίνει επειδή ο σειριακός FW-Recursive τρέχει πιο γρήγορα από τον σειριακό FW.

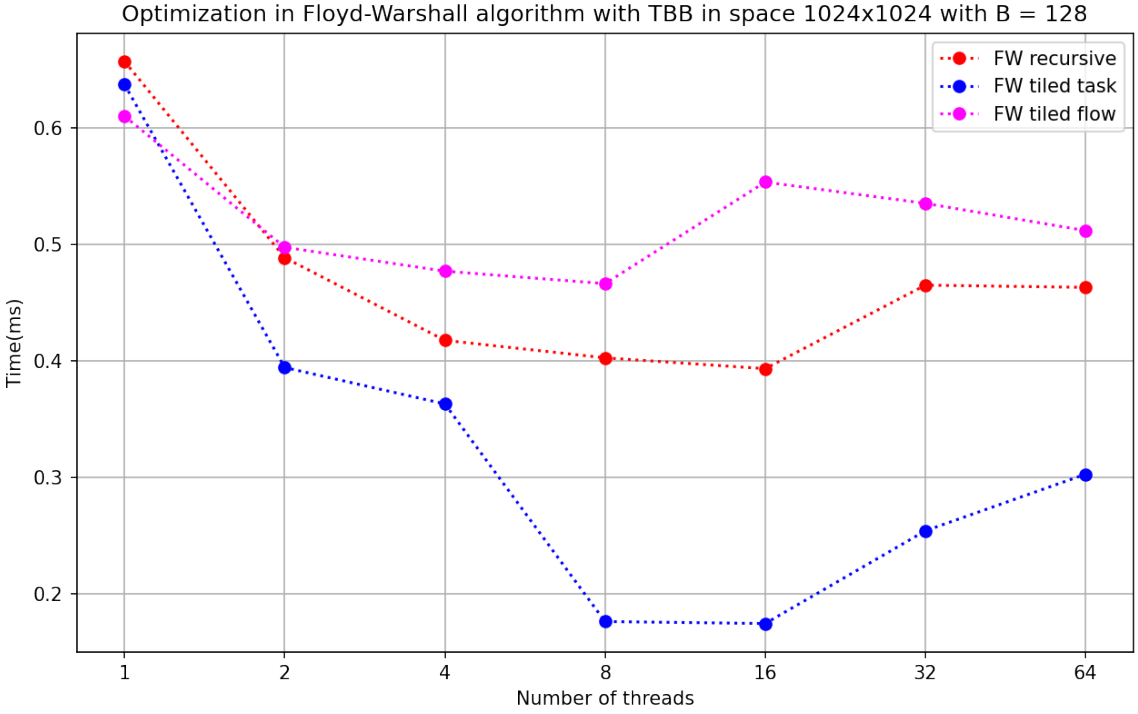
- FW-Tiled: Υλοποιήσαμε τον FW-Tiled με χρήση parallel for και των κατάλληλων barriers για τον σωστό συγχρονισμό των threads. Παρατηρούμε ότι η συγκεκριμένη υλοποίηση του αλγόριθμου κλιμακώνει αρκετά καλά και φτάνει μέχρι και 20 speedup για 64 threads. Πιθανότατα δεν μπορούμε να επιτύχουμε παραπάνω λόγω του νόμου του Amdahl (δεν είναι όλο το πρόγραμμα παράλληλο), λόγω του χρόνου που χάνεται όταν πρέπει να συγχρονιστούν τα threads καθώς και του χρόνου που χάνεται όταν κάποιο thread έχει cache miss και πρέπει να φέρει την τιμή από τη μνήμη ή από την cache άλλου επεξεργαστή.

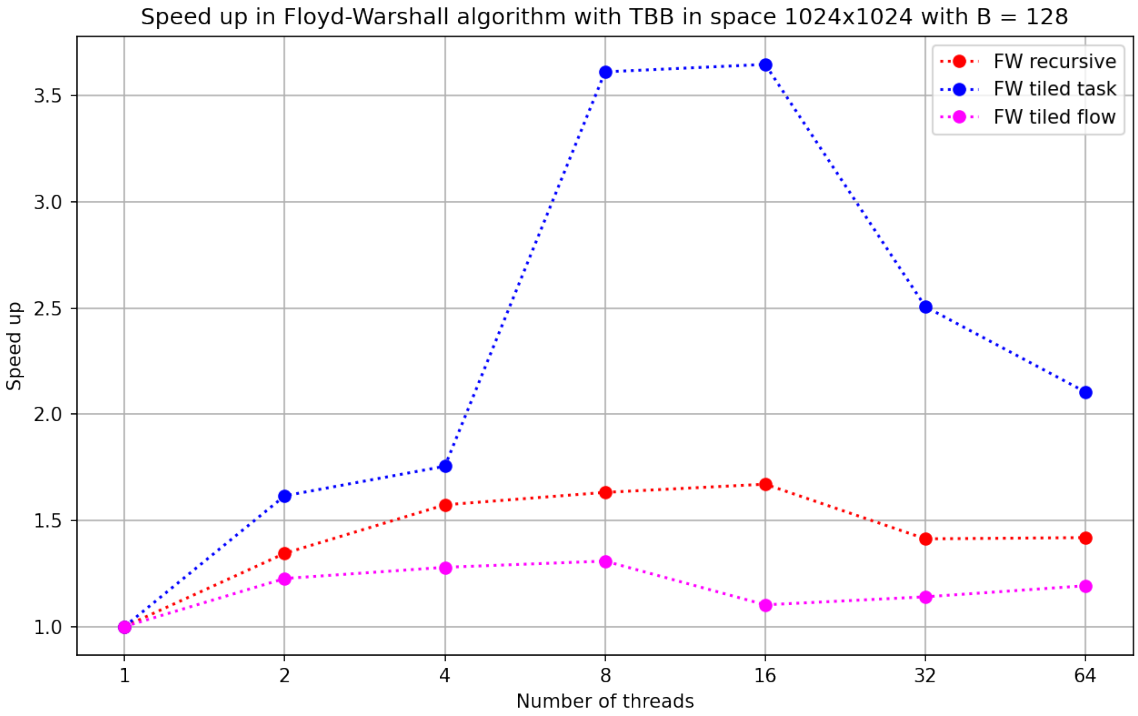
3 Threading Building Blocks

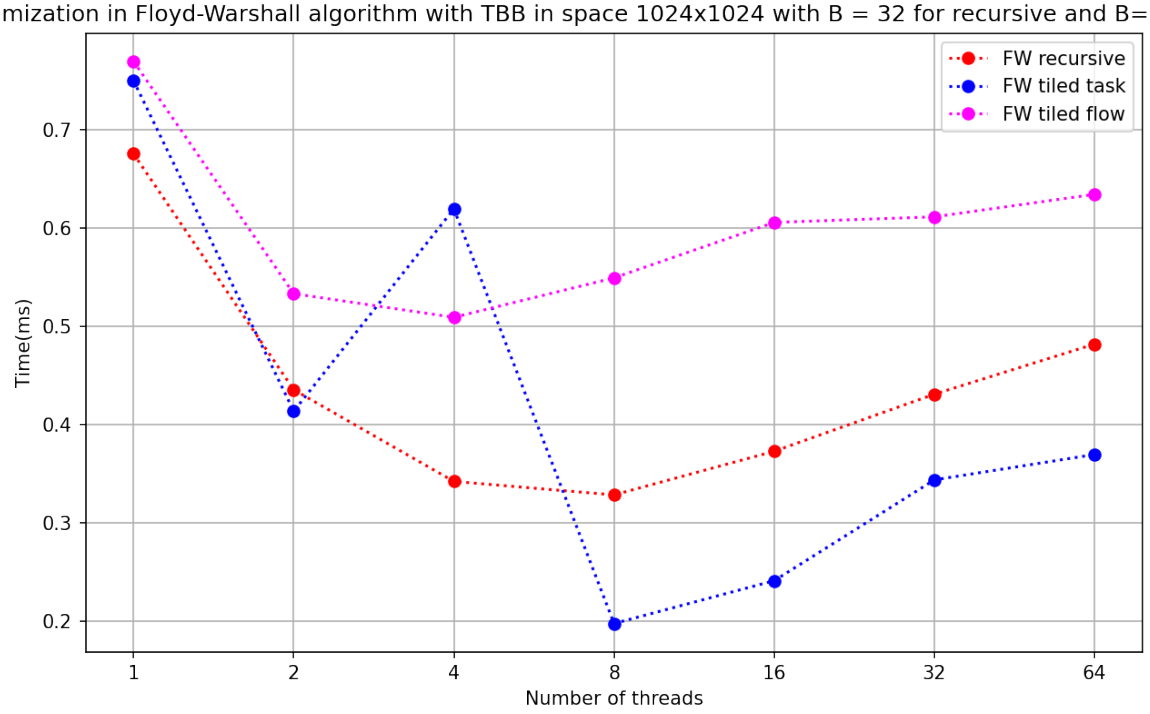
3.1 Size 1024x1024

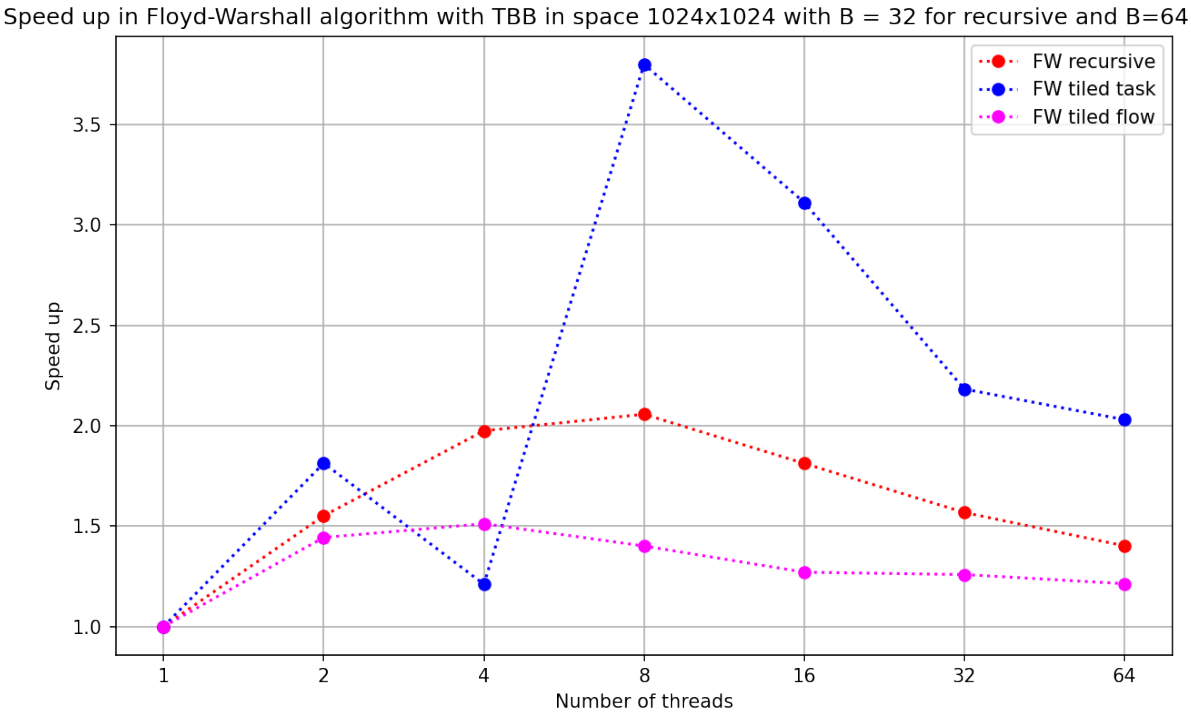



Speed up in Floyd-Warshall algorithm with TBB in space 1024x1024
FW
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1
2
4
8
16
32
64
1
2
3
4
5
6
7
8
9

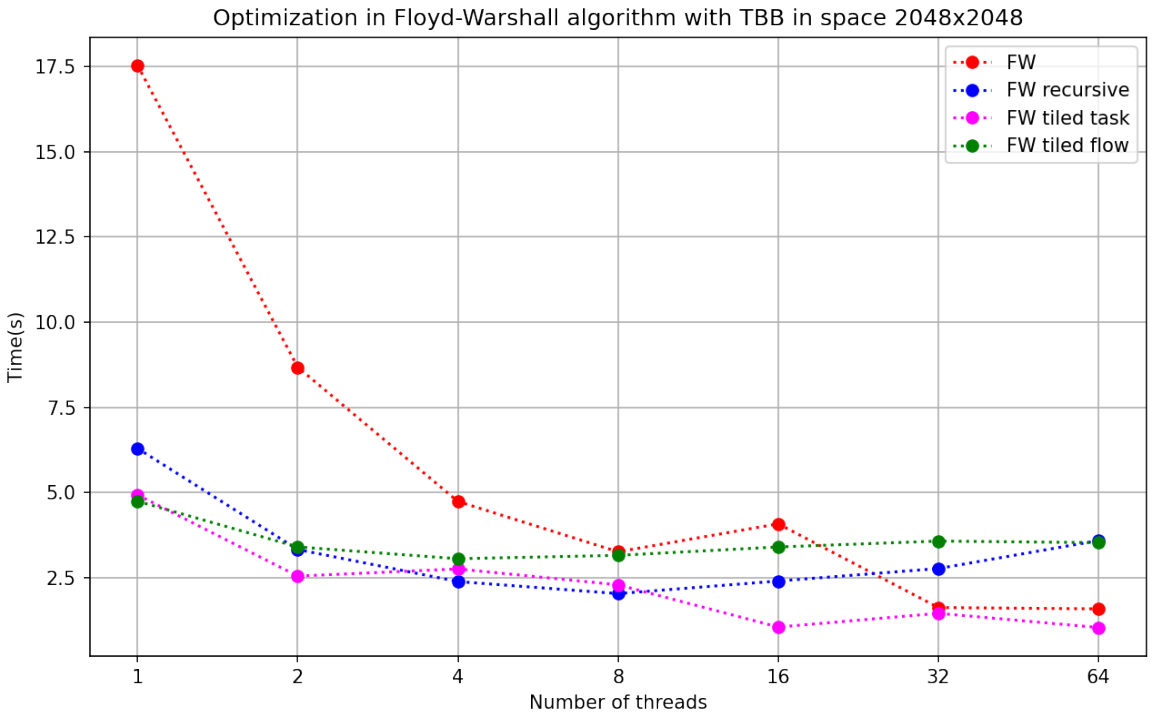

Optimization in Floyd-Warshall algorithm with TBB in space 1024x1024 with B = 128
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
1
2
4
8
16
32
64
0.2
0.3
0.4
0.5
0.6

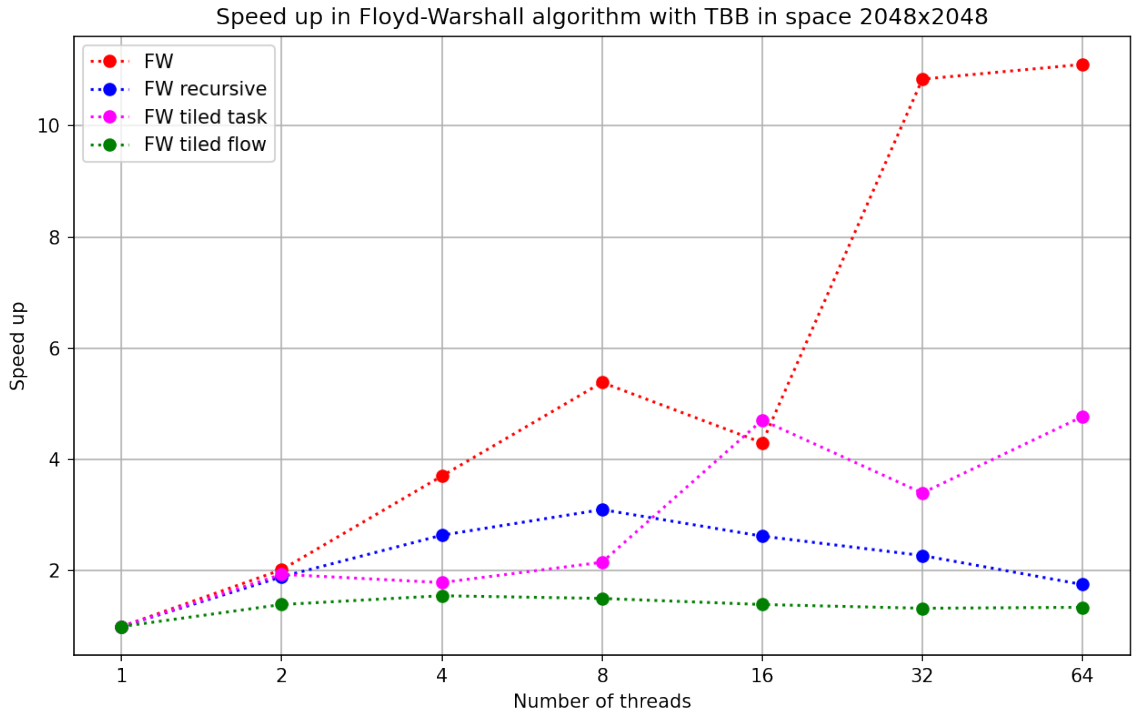

Speed up in Floyd-Warshall algorithm with TBB in space 1024x1024 with B = 128
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1.0
1.5
2.0
2.5
3.0
3.5
1
2
4
8
16
32
64

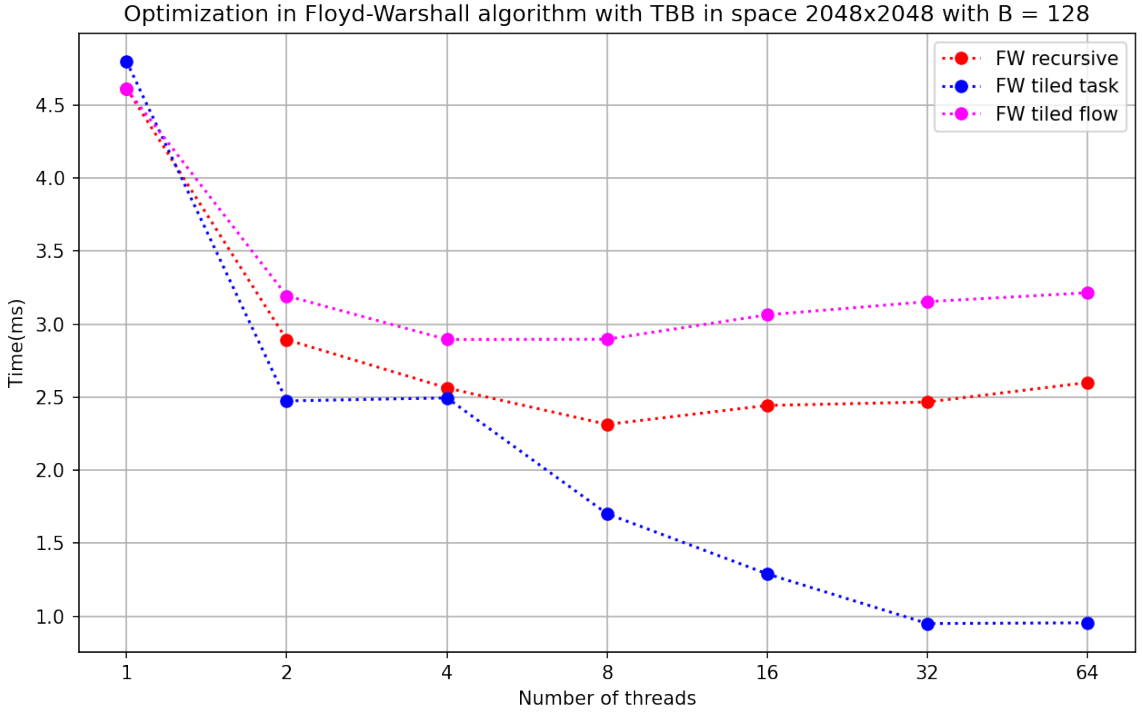

Optimization in Floyd-Warshall algorithm with TBB in space 1024x1024 with B = 32 for recursive and B=64 for tiled
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
0.2
0.3
0.4
0.5
0.6
0.7
1
2
4
8
16
32
64

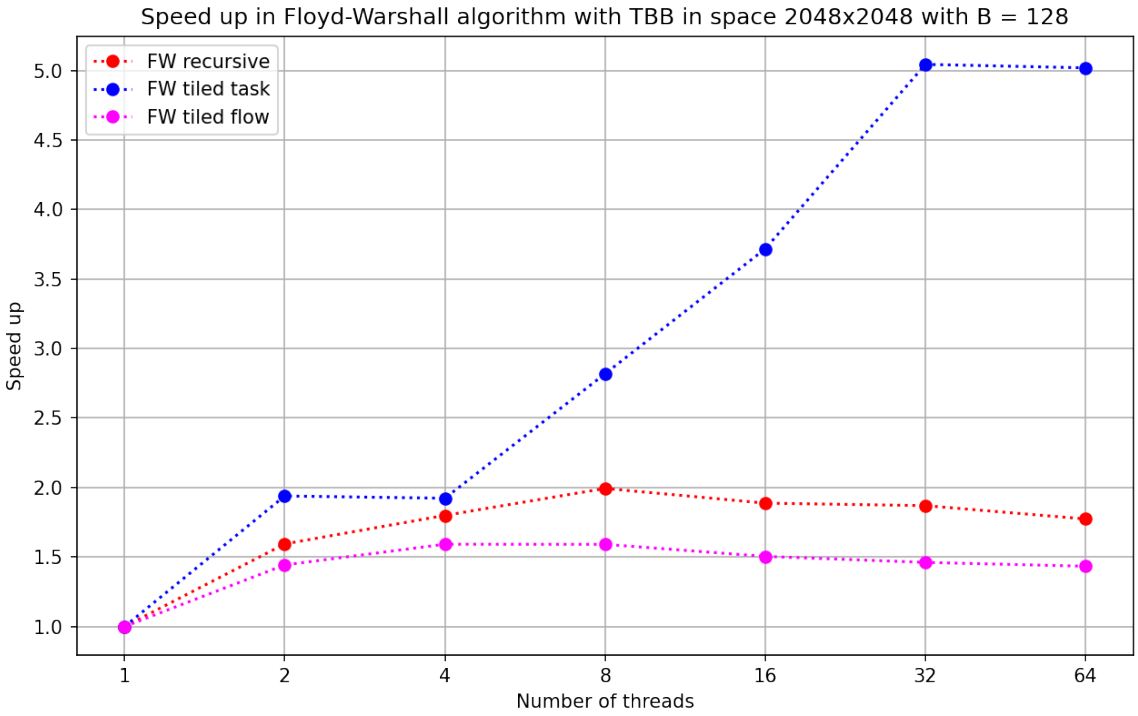


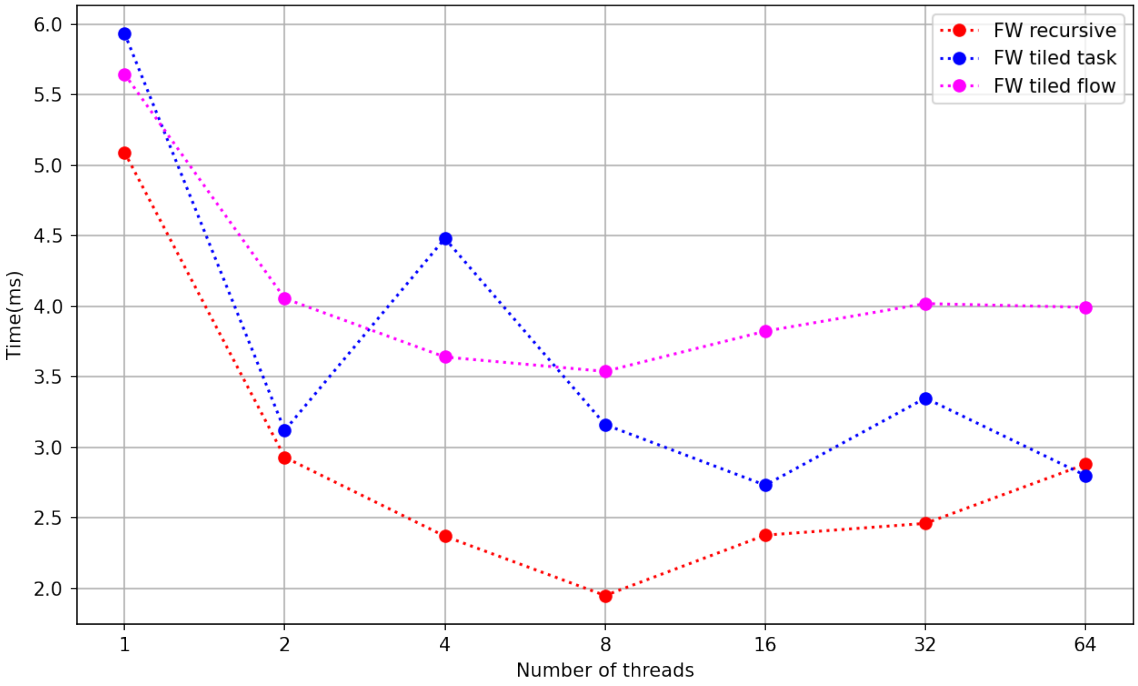
3.2 Size 2048x2048

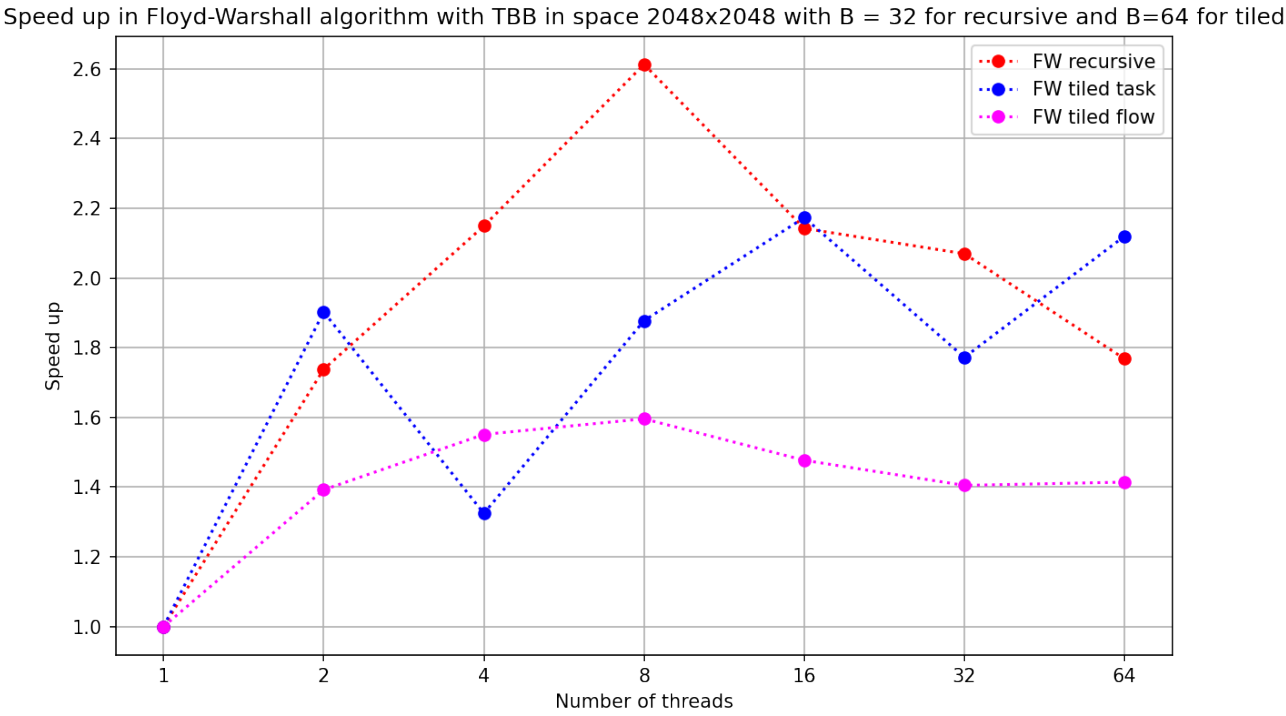



Speed up in Floyd-Warshall algorithm with TBB in space 2048x2048
FW
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1
2
4
8
16
32
64
2
4
6
8
10

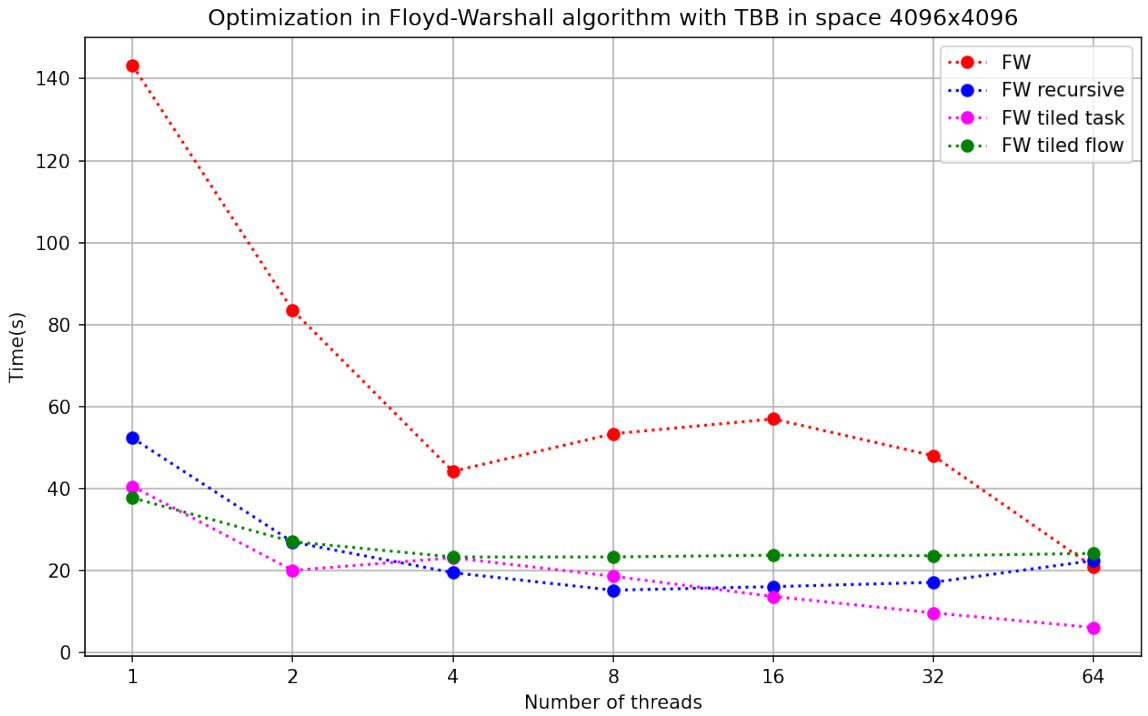

Optimization in Floyd-Warshall algorithm with TBB in space 2048x2048 with B = 128
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
1
2
4
8
16
32
64
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

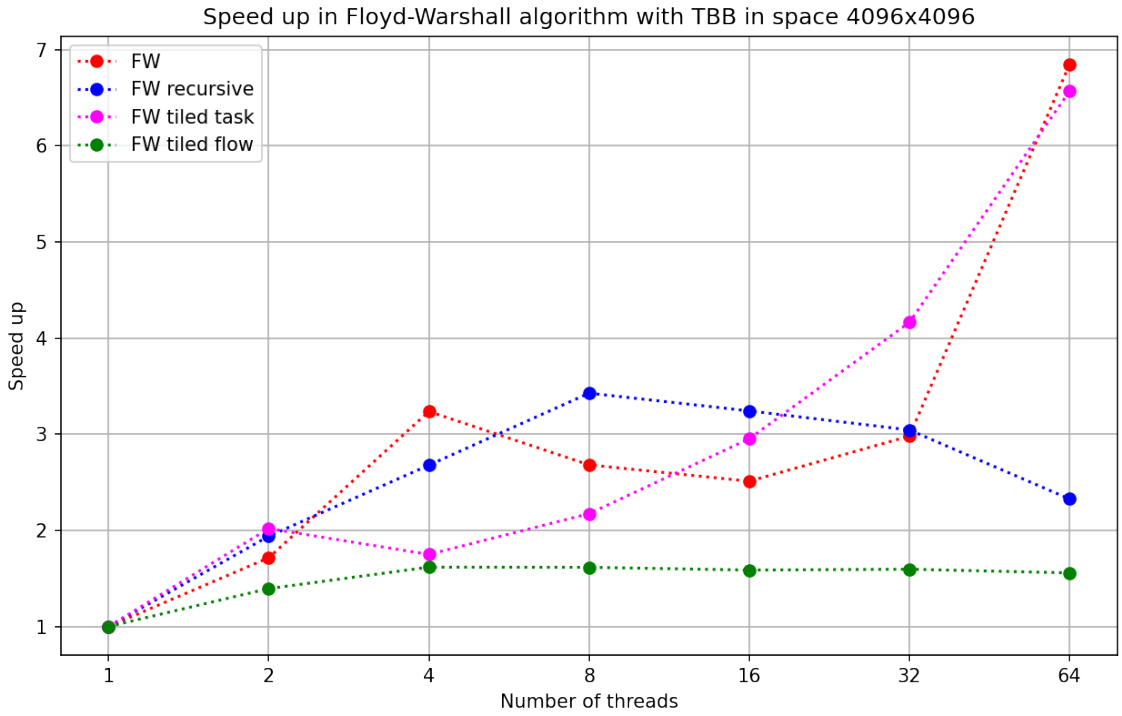

Speed up in Floyd-Warshall algorithm with TBB in space 2048x2048 with B = 128
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
1
2
4
8
16
32
64

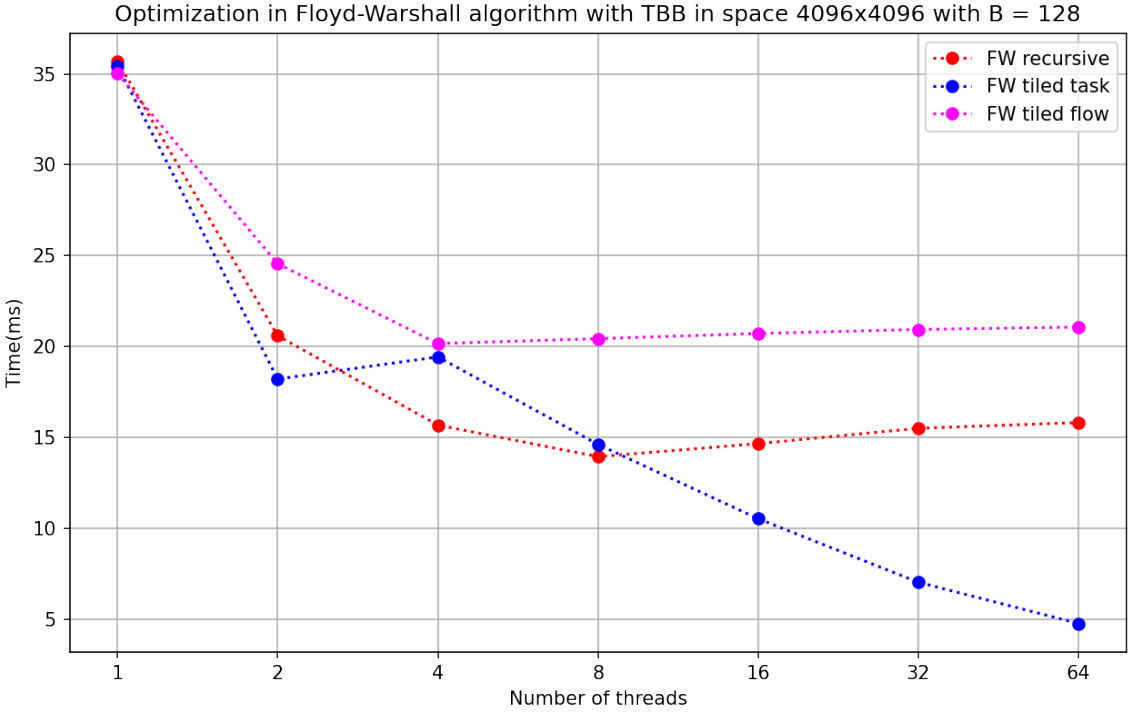

Optimization in Floyd-Warshall algorithm with TBB in space 2048x2048 with B = 32 for recursive and B=64 for tiled
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
1
2
4
8
16
32
64

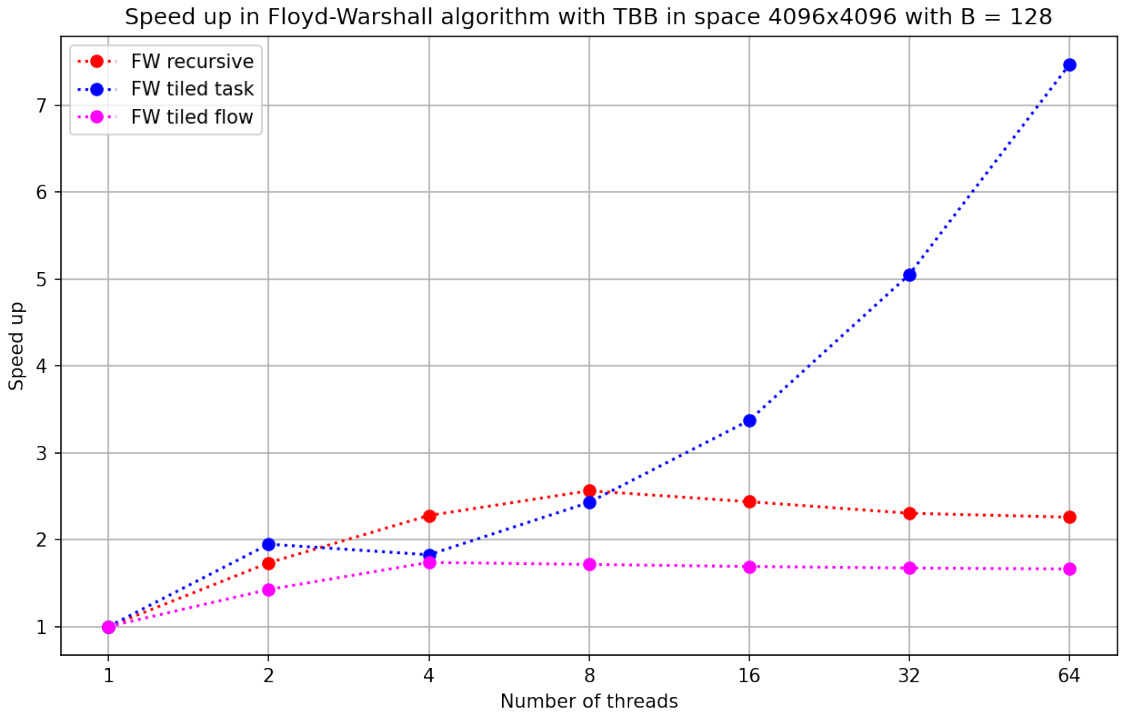


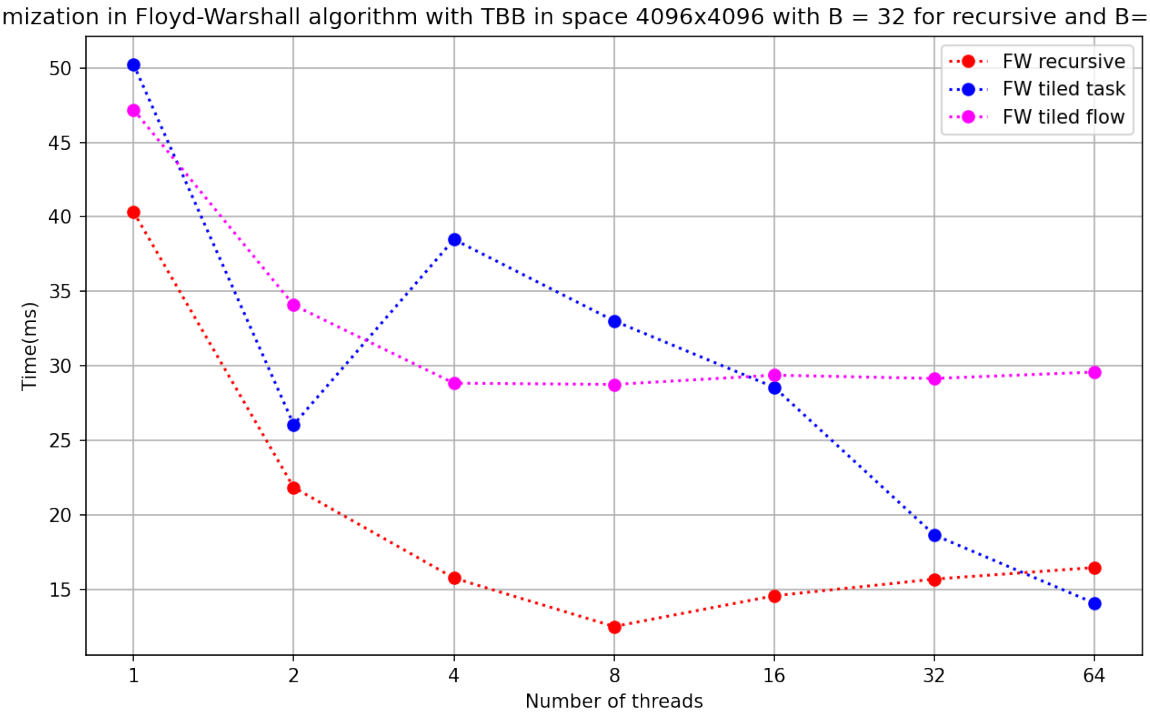
3.3 Size 4096x4096

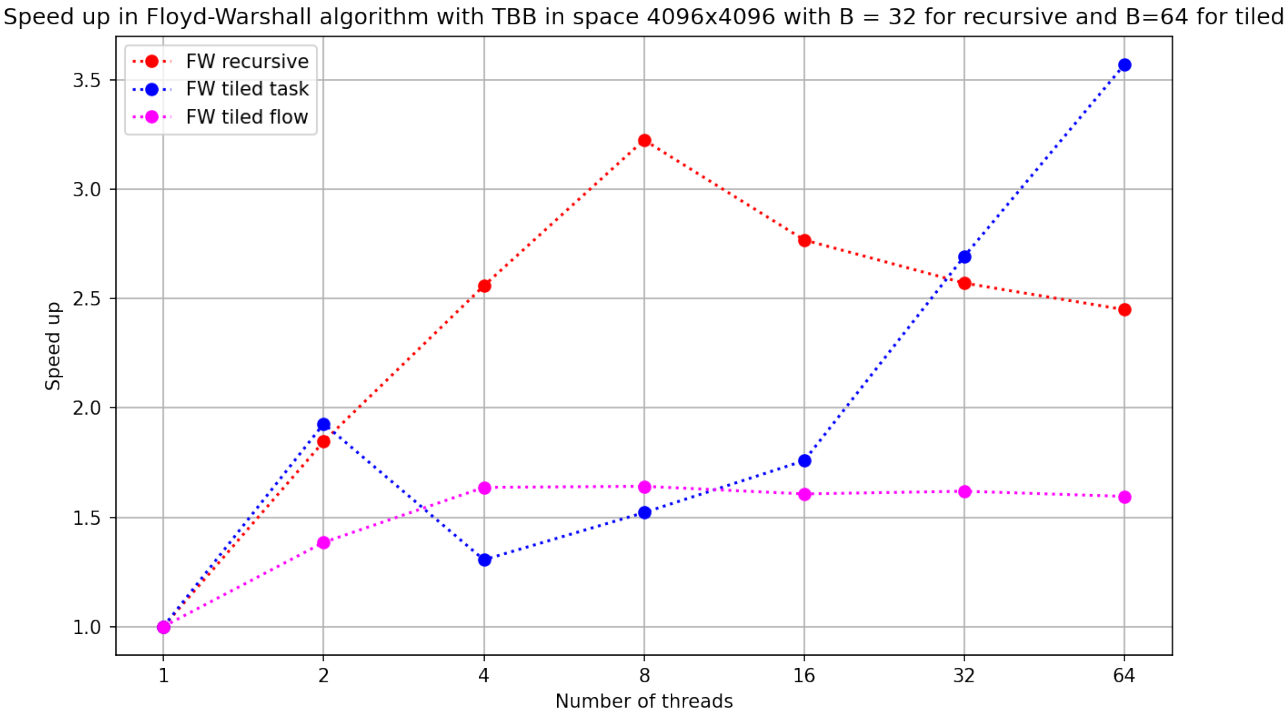



Speed up in Floyd-Warshall algorithm with TBB in space 4096x4096
FW
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1
2
4
8
16
32
64
1
2
3
4
5
6
7


Optimization in Floyd-Warshall algorithm with TBB in space 4096x4096 with B = 128
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
1
2
4
8
16
32
64
5
10
15
20
25
30
35


Speed up in Floyd-Warshall algorithm with TBB in space 4096x4096 with B = 128
FW recursive
FW tiled task
FW tiled flow
Speed up
Number of threads
1
2
3
4
5
6
7
1
2
4
8
16
32
64


Optimization in Floyd-Warshall algorithm with TBB in space 4096x4096 with B = 32 for recursive and B=64 for tiled
FW recursive
FW tiled task
FW tiled flow
Time(ms)
Number of threads
15
20
25
30
35
40
45
50
1
2
4
8
16
32
64



Παρατηρήσεις (κυρίως για size 4096x4096):

- FW: Παρατηρούμε ότι ο απλός FW αλγόριθμος δεν κλιμακώνει καλά παρότι στην TBB υλοποίηση φροντίζουμε για το thread affinity. Παρατηρούμε καλό speedup μέχρι τα 4 threads, μετά όμως το speedup μειώνεται (πιθανώς λόγω της αρχιτεκτονικής του sandman και της ανάγκης για επικοινωνία μεταξύ των threads) και τελικά επιτυγχάνεται ένα πολύ μέτριο speedup 7 για 64 threads.

- FW-Recursive: Υλοποιήσαμε σε TBB έναν task-based FW-Recursive. Βλέπουμε ότι το μέγιστο speedup επιτυγχάνεται για 8 threads και μετά το scalability "σπάει". Αυτό πιθανότατα σημαίνει ότι στην task-based υλοποίηση με ένα thread pool που αναλαμβάνει τα tasks, όταν έχουμε παραπάνω απο 8 threads η επικοινωνία κοστίζει περισσότερο από την ωφελιμότητα της παραλληλίας. Αξίζει να σημειωθεί ότι το speedup είναι μικρότερο από τον απλό FW επειδή ο σειριακός FW-Recursive εκτελείται γρηγορότερα από τον σειριακό FW και έτσι έχουμε μικρότερο αριθμητή.

- FW-Tiled:

  - Task-Based: Υλοποιήσαμε μια task εκδοχή του FW-Tiled σε TBB. Το speedup είναι όμοιο σε εικόνα με αυτό του απλού FW, επομένως περιμένουμε να δικαιολογείται ομοίως.
  - Flowgraph: Υλοποιήσαμε επίσης μια εκδοχή του FW-Tiled με βάση το flowgraph του (το οποίο είχαμε παραθέσει στην ενδιάμεση αναφορά). Παρατηρούμε ότι η υλοποίηση αυτή δεν κλιμακώνει καθόλου μετά τα 2 threads. Πιστεύουμε ότι αυτό οφείλεται στο ότι το flowgraph module των TBBs δεν εξασφαλίζει καθόλου καλό load balancing και έτσι τα παραπάνω threads δεν χρησιμοποιούνται σχεδόν καθόλου.

4 Βελτιστοποίηση του tiled αλγορίθμου

Τέλος, εφόσον φαινόταν ότι η FW-Tiled υλοποίηση του αλγορίθμου σε OpenMP είναι σαφώς ταχύτερη από τις υπόλοιπες, προσπαθήσαμε να την βελτιστοποιήσουμε όσο το δυνατόν περισσότερο. Έτσι, χρησιμοποιώντας parallel for με nowait και τα κατάλληλα barriers, και θέτοντας τα OpenMP flags **OMP_PLACES=threads**, **OMP_PROC_BIND=close** πετύχαμε βέλτιστο χρόνο 1.4907 seconds. Με αυτά τα flags καταφέρνουμε να ορίσουμε καλύτερα την περιοχή εργασίας κάθε thread και να διατηρήσουμε την τοπικότητα. Έτσι ακόμη και όταν ένα thread έχει cache miss η τιμή πιθανότατα θα βρίσκεται σε κάποιο κοντινό επεξεργαστή και θα βρίσκεται σε μικρή απόσταση ώστε να ανακτηθεί γρήγορα. Η ορθότητα ελέγχθηκε τρέχοντας τον σειριακό αλγόριθμο για ίδια αρχικοποίηση και βλέποντας ότι παράγουν ίδιο τελικό πίνακα.