A.M: 1115201600102,
1115201400190
ΟΝΟΜΑΤΑ: ΜΗΤΣΙΟΣ ΑΛΕΞΑΝΔΡΟΣ,
ΣΤΑΜΕΛΙΑΣ ΠΡΟΚΟΠΙΟΣ

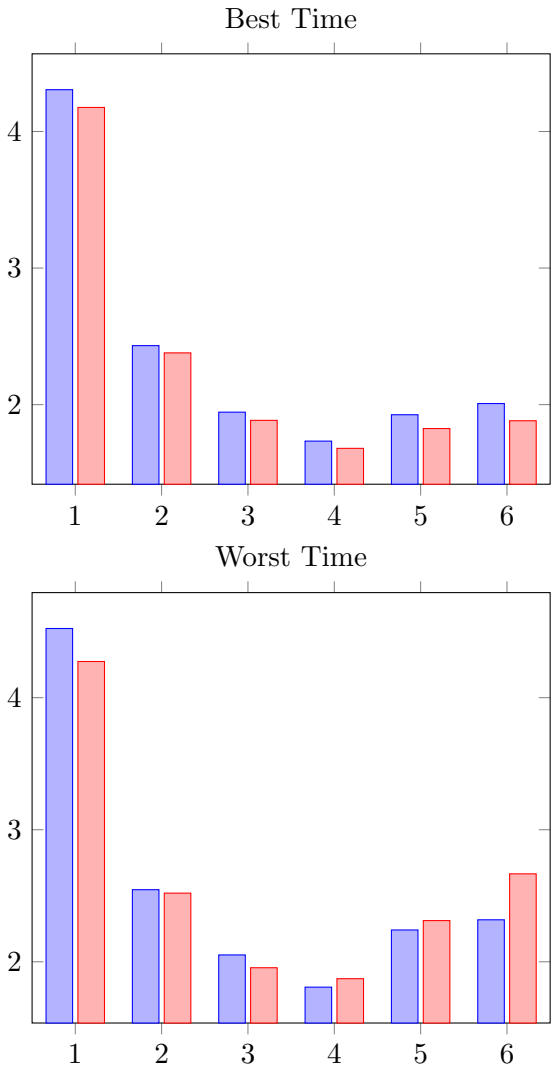
Προβλήματα που συναντήσαμε κατά την υλοποίηση:
Κατά τη διάρκεια της υλοποίησης για τη παραλληλοποίηση της αναζήτησης αντιμετωπίσαμε το εξής θέμα.Ο Job Counter έπρεπε να προστατευτεί με mutex διότι η αύξηση ή η μείωση του επηρέαζε τη συνάρτηση GetNextAvailRes με αποτέλεσμα κάποιες φορές να έχουμε seg fault ή τη εμφάνιση λάθος αποτελέσματων.
Επίσης η χρήση mutex(στο Job Counter) μας βοήθησε σε αυτή τη φάση της υλοποίησης στο να κάνουμε τη κατάλληλη στιγμή με χρήση if την εντολή pthreadsignal στο αντίστοιχο condition variable το οποίο χρησίμευε ώστε η getNextAvailRes να αρχίσει να τρέχει όταν δεν θα υπάρχουν MatchDocument δουλείες μέσα στην ουρά.
Γενικά η παραλληλοποίηση της search ΜΟΝΟ, δεν ήταν δύσκολη διαδικασία καθώς δεν υπήρχαν πολλές κοινές δομές που έπρεπε να προστατευθούν με mutex.Επίσης στη συγκεκριμένη φάση της παραλληλοποίησης έπρεπε να προστατευτεί μία άλλη δομή με χρήση mutex, η δομή στοίβας για τα αποτελέσματα, καθώς η πρόσβαση από παραπάνω από ένα νήματα εμπόδιζε τη σωστή εισαγωγή των αποτελεσμάτων.
Κατά τη διάρκεια της υλοποίησης για τη παραλληλοποίηση εισαγώγής,διαγραφής και search, τα θέματα που αντιμετωπίσαμε ήταν πολλά, καθώς όχι μόνο χρειάστηκε να προστατευτούν οι κοινές δομές με χρήση mutex , αλλά για τη ορθή εκτέλεση του προγράμματος χρειάστηκε να μπαίνουν στη ουρά αναμονής του JobschedulerNode ¨ίδιες' δουλείες καθώς δεν ήταν σίγουρο ότι αν μια δουλεία μπει πρώτη θα τελειώσει και πρώτη από κάποια άλλη και έτσι υπήρχε ο κίνδυνος να εκτελεστεί μία δουλεία MatchDocument χωρίς να έχει τελειώσει μία StartQuery που να τρέχει τη συγκεκριμένη χρονική στιγμή και αντίστροφα.
Έτσι με χρήση condition variable λύσαμε το συγκεκριμένο θέμα στη ουρά με τις δουλείες υπήρχαν μόνο ¨ίδιες' εργασίες.
Επίσης, άλλο ένα θέμα που χρειάστηκε να δούμε είναι η σωστή χρονικά ανάθεση κάθε δουλειάς στην ουρά, δηλαδή οι συναρτήσεις StartQuery, EndQuery, MatchDocument να τσεκάρουν με χρήση κατάλληλων μετρητών αν έχουν τελειώσει δουλειές άλλης ¨κατηγορίας¨.
Επιπλέον, ένα ακόμα corner case της συγκεκριμένης παραλληλοποίησης που έπρεπε να αντιμετωπιστεί είναι να έχει μπει εργασία τύπου StartQuery και στη συνέχεια EndQuery και μόλις η EndQquery τσεκάρει με το if αν υπάρχει α-

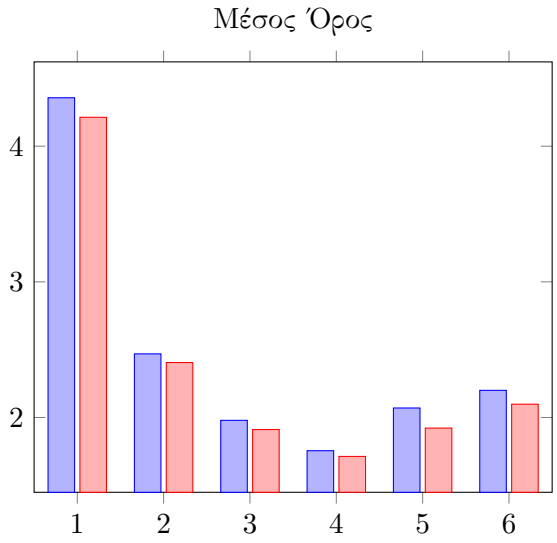
ντίστοιχη δουλειά, να γίνεται από τη τελευταία δουλειά της StartQuery signal και να μη προλαβάινει το κεντρικό thread να πάει στη wait , το συγκεκριμένο θέμα λύθηκε με χρήση μιας if condition.
Δοκιμάσαμε ,επίσης και υλοποιήσαμε την παραλληλοποίηση και των StartQuery, EndQuery , αλλά όπως αποδεικνύεται και από τα παρακάτω διαγράμματα και τους πίνακες, η παραλληλοποίησή τους αύξανε, έστω και ελάχιστα τον χρόνο εκτέλεσης.
Παρακάτω παρατίθενται οι πίνακες και τα διαγράμματα με πληροφορίες για την εκτέλεση.(Για την κάθε περίπτωση χρησιμοποιήθηκε δείγμα 100 εκτελέσεων).

All Parallelized			
Αριθμός Threads	Καλύτερος Χρόνος	Χειρότερος Χρόνος	Μέσος Όρος
1	4.306 sec	4.524 sec	4.357 sec
2	2.432 sec	2.546 sec	2.469 sec
3	1.945 sec	2.052 sec	1.979 sec
4	1.733 sec	1.808 sec	1.755
5	1.926 sec	2.241 sec	2.070 sec
6	2.008 sec	2.318 sec	2.200 sec

Search Parallelized			
Αριθμός Threads	Καλύτερος Χρόνος	Χειρότερος Χρόνος	Μέσος Όρος
1	4.176 sec	4.274 sec	4.213 sec
2	2.379 sec	2.520 sec	2.405 sec
3	1.885 sec	1.955 sec	1.911 sec
4	1.680 sec	1.872 sec	1.713 sec
5	1.825 sec	2.312 sec	1.992 sec
6	1.882 sec	2.666 sec	2.098 sec

pink: Search Parallelized
blue: All Parallelized


Best Time
4
3
2
1
2
3
4
5
6
Worst Time
4
3
2
1
2
3
4
5
6


Μέσος Όρος
4
3
2
1
2
3
4
5
6