ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένος μετρητής πλήθους με χρήση map reduce

Task 1.2

Διδάσκουσα: Ε. Κατσίρη

Γιαννακόπουλος Γιώργιος (Α.Μ 6167)
Θεοδοσιάδου Γεωργία (Α.Μ. 470 ΜΔΕ)

Σκοπός του συγκεκριμένου τμήματος της εργασίας είναι η ποιοτική περιγραφή του σκελετού του client, τον οποίο βάσει του σεναρίου πάνω στο οποίο εργαζόμαστε τον αντιμετωπίζουμε ως έναν από τους workers του κατανεμημένου συστήματός μας. Πριν περάσουμε στην αναλυτική παρουσίαση του διαγράμματος ροής και της δήλωσης των διαφόρων συναρτήσεων και δομών δεδομένων, θα πρέπει πρώτα να αναφερθούμε στα βασικά στάδια λειτουργίας από τα οποία περνάει ο client μας. Τα στάδια αυτά παρουσιάζονται παρακάτω.

Φάση 1η : *init()*

Είναι το πρώτο στάδιο από το οποίο περνάει ο κάθε worker και στο οποίο γίνεται σάρωση του χώρου και καταγραφή των ετικετών (tags) που βρίσκονται μέσα στον χώρο, δηλαδή μια καταγραφή των ανθρώπων που υπάρχουν στον υπό σάρωση χώρο. . Οι ετικέτες αυτές περιλαμβάνουν ένα μοναδικό αναγνωριστικό καθώς επίσης και το φύλο του εκάστοτε επισκέπτη. Για να ξεκινήσει αυτή η διαδικασία θα πρέπει ο client να λάβει το μήνυμα *runInit()* από τον Master. Ο κάθε worker στο σημείο αυτό δημιουργεί μια τοπική δομή δεδομένων, στην συγκεκριμένη περίπτωση ένα dictionary,το οποίο περιλαμβάνει τα παραπάνω στοιχεία για όλους τους παρευρισκόμενους επισκέπτες. Με το πέρας αυτής της διαδικασίας αποστέλλεται στον Master το μήνυμα *Init_OK*.

Φάση 2η : *Map()*

Ο Master λαμβάνοντας το μήνυμα *Init_OK* γνωρίζει ότι πλέον ο client μας είναι σε θέση να προχωρήσει στην επόμενη κατάσταση. Αποστέλλει επομένως το μήνυμα *Run_Map()* το οποίο ενημερώνει τον client ότι μπορεί να ξεκινήσει την διαδικασία του mapping. Στα πλαίσια αυτής φάσης ο worker δημιουργεί μια νέα τοπική δομή δεδομένων η οποία περιλαμβάνει μόνο το φύλο του εκάστοτε επισκέπτη, αφαιρούνται δηλαδή τα πεδία που περιλαμβάνουν τα αναγνωριστικά των ετικετών. Τελειώνοντας αυτή την διεργασία αποστέλλεται ένα μήνυμα *Map_OK* στον Master.

Φάση 3η : *Shuffle()*

Ο Master λαμβάνοντας αυτή τη φορά το μήνυμα *Map_OK* δίνει την εντολή στο client να ξεκινήσει την διαδικασία Shuffle στέλνοντας αυτήν την φορά το μήνυμα *Run_Shuffle()*. Στο σημείο αυτό ο client μέσω υποδοχών (shockets) στέλνει στον Master την δομή δεδομένων που αποθήκευσε τοπικά στο προηγούμενο επίπεδο. Επιπλέον για να ενημερώσει τον Master ότι η διαδικασία ολοκληρώθηκε με επιτυχία στέλνει και το μήνυμα *Shuffle_OK*.

Φάση 4η : *PrepareReduceInput()*

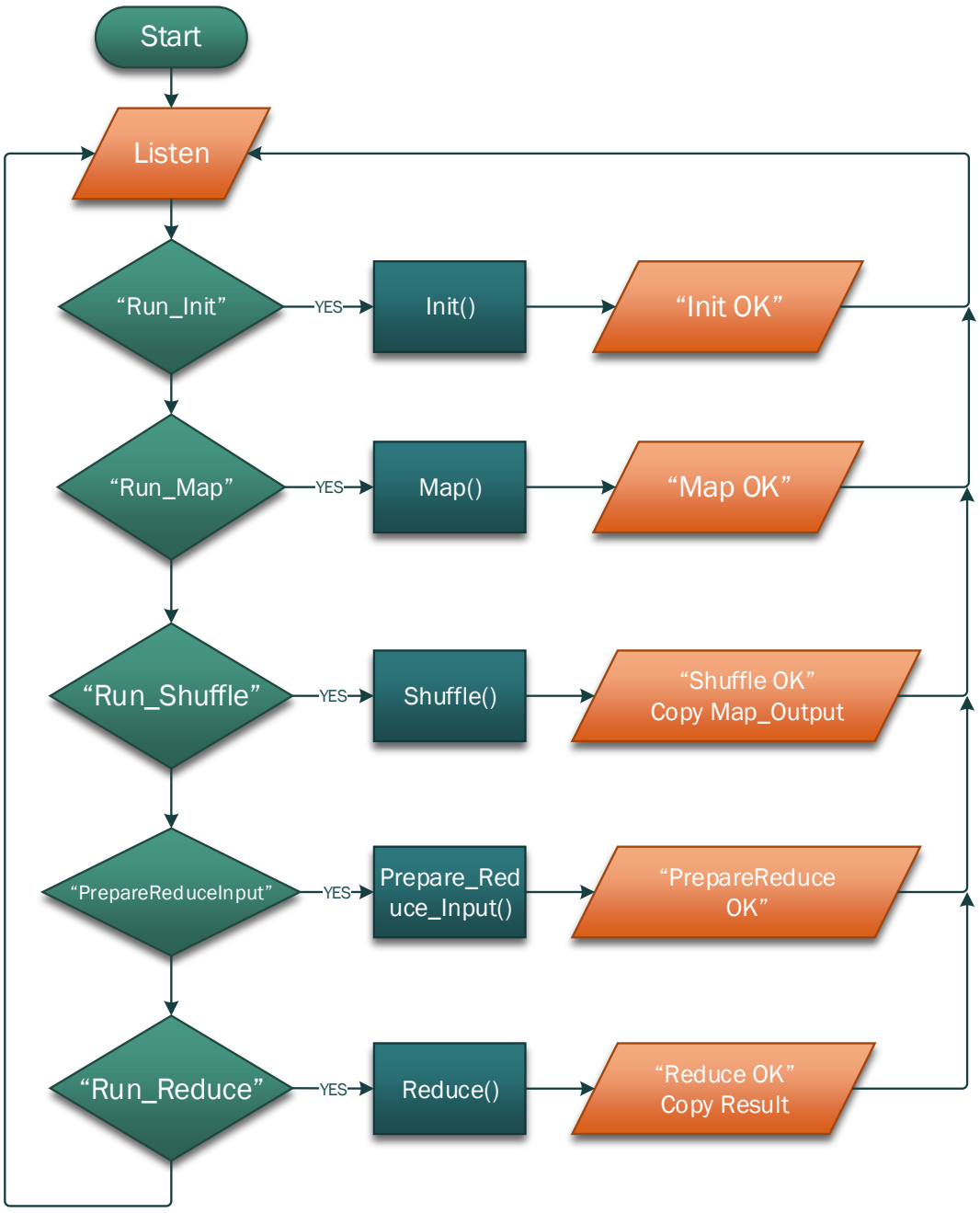
Έχοντας λάβει τις δομές δεδομένων από όλους τους workers, ο Master δημιουργεί μια ενιαία δομή που τις περιλαμβάνει όλες. Στην συνέχεια η δομή αυτή διαιρείται σε ένα αριθμό επιμέρους τμημάτων (shards) τα οποία μοιράζονται εκ νέου στους worker ούτως ώστε να επεξεργαστούν επιμέρους από τον καθένα. Έτσι στην φάση αυτή, σε επίπεδο client, έχουμε λήψη των δεδομένων που στέλνονται από τον Master μέσω υποδοχών και αποθήκευσή τους σε μια τοπική δομή δεδομένων. Φυσικά, και σε αυτό το σημείο στέλνεται το αντίστοιχο μήνυμα επιβεβαίωσης (*PrepareReduceInput_OK)* στον Master.

Φάση 5η : *Reduce()*

Τέλος, έχοντας λάβει το μήνυμα *Run_Reduce* ο client συνενώνει τα ζεύγη <man,1> υπολογίζοντας τον συνολικό αριθμό των αντρών. Η αντίστοιχη διαδικασία

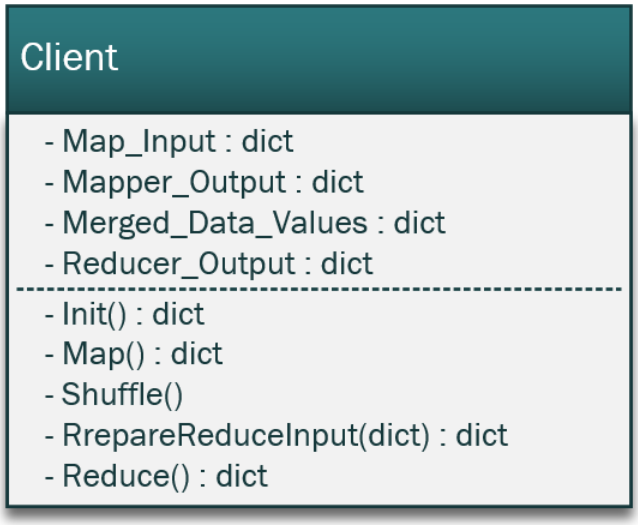
επαναλαμβάνεται και για τις γυναίκες. Αποθηκεύει αρχικά τα αποτελέσματα σε μια τοπική δομή και στην συνέχεια την στέλνει στον Master μέσω υποδοχών.

Όλα όσα αναφέρθηκαν παραπάνω εμφανίζονται στο διάγραμμα ροής του σχήματος 1. Παρατηρούμε ότι ο client βρίσκεται σε μια μόνιμη κατάσταση αναμονής περιμένοντας να λάβει κάποιο μήνυμα από τον client. Ανάλογα με το μήνυμα το οποίο θα λάβει προβαίνει και στην υλοποίηση της αντίστοιχης φάσης. Τέλος σε κάθε φάση επιστρέφει το αντίστοιχο μήνυμα επιβεβαίωσης, καθώς και τις αντίστοιχες δομές δεδομένων όπου αυτό είναι αναγκαίο, και επιστρέφει στην κατάσταση αναμονής περιμένοντας το επόμενο μήνυμα.



Σχήμα 1. Διάγραμμα Ροής Client

Στην συνέχεια στο σχήμα 2 παρουσιάζεται η δομή του client μας σε UML. Καθώς πρόκειται για μια μεμονωμένη κλάση θα έχουμε ένα μόνο στοιχείο το οποίο περιλαμβάνει τις τοπικές δομές δεδομένων της υπό μελέτη κλάσης καθώς και τις συναρτήσεις που της αντιστοιχούν. Στο πάνω μισό γίνεται η δήλωση των δομών ενώ στο κάτω έχουμε την δήλωση των συναρτήσεων.



Σχήμα 2. UML κλάσης Client

Σε τελικό στάδιο γίνεται η δήλωση των συναρτήσεων που παρουσιάστηκαν παραπάνω, σχήμα 3. Οι συναρτήσεις *ReduceInput* και *Output* προέρχονται από τον κώδικα mapreduce.py που δόθηκε στα πλαίσια του μαθήματος.

```
Class Client:
    def __init__(self)
    self.Map_Input={}
    self.Mapper_Output={}
    self.Merged_data_values={}
    self.Reducer_output={}

    def initialize(self):
        self.Map_Input.update(scan())

    def map(self):
        self.Mapper_Output.update(ReduceInput(Map_Input))

    def Shuffle(self):
        Master.shuffled_data[shard_id].update(self.Mapper_Output)

    def PrepareReduceInput(self,merged):
        self.Merged_data_values.update(merged)

    def Reduce(self):
        self.Reducer_output.update(Output(Merged_data_values))

}
```

Σχήμα 3. Δήλωση συναρτήσεων σε Python