Κ 24

Homework 2: Ημερομηνία παράδοσης: 2/6/2010

Σημείωση: 1) Για τα προγράμματα που θα γράψετε, παρακαλείστε να παρουσιάσετε ένα σύντομο αρχείο README που να εξηγεί τη λογική του προγράμματος σας. Επίσης πρέπει να συνοδεύονται με ένα Makefile που να αναλαμβάνει το «χτίσιμο» του προγράμματος σας. Οδηγίες για την υποβολή της άσκησης υπάρχουν στη σελίδα του μαθήματος.

Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις και βαθμολόγηση, κλπ) είναι: Nikos Chondros (n.chondros@di.uoa.gr) and Christos Patsonakis (std03193@di.uoa.gr).

Άσκηση 1: Multi-threaded MapReduce (50 points). Γράψτε έναν πολυνηματικό πρόγραμμα που ονομάζεται mapReduceThreaded και υλοποιεί τον MapReduce αλγόριθμο για τον οποίο μάθατε στη δεύτερη εργασία. Το πρόγραμμα θα το τρέχετε από τη γραμμή εντολής με τα ακόλουθα ορίσματα:

```
prompt> mapReduceThreaded [numMappers] [numReducers] [bufferSize]
[input_folder] [output_folder]
```

Συγκεκριμένα, τα ορίσματα είναι:

- **numMappers:** ο αριθμός νημάτων mappers που θα δημιουργήσει το πρόγραμμα
- **numReducers:** ο αριθμός νημάτων reducers που θα δημιουργήσει το πρόγραμμα
- **bufferSize:** το μέγεθος των ενταμιευτών που θα κρατάνε δεδομένα που στέλνονται από τους mappers στους reducers. Πρέπει να είναι > 0.
- **input_folder:** το όνομα του κατάλογου που περιέχει τα αρχεία εισόδου
- **output_folder**: το όνομα του κατάλογου όπου θα τοποθετούνται τα αρχεία εξόδου

Το πρόγραμμα θα δημιουργεί *numMappers* νήματα για να παίξουν τον ρόλο των mappers και *numReducers* νήματα για να παίξουνε τον ρόλο των reducers. Τα νήματα mappers και reducers θα συμπεριφέρονται όπως οι αντίστοιχες διεργασίες που δημιουργήσατε στην δεύτερη εργασία με διαφορά στην επικοινωνία μεταξύ τους. Αντί για named pipes, η επικοινωνία θα γίνεται μέσω ενταμιευτών. Κάθε νήμα reducer θα έχει έναν ενταμιευτή συγκεκριμένου μεγέθους (που ορίζεται από το *bufferSize*) από τον οποίο θα διαβάζει ζευγάρια (key, value) pairs που θα στέλνουν τα νήματα mappers σε αυτό το νήμα reducer για επεξεργασία. Οπότε, στο πρόγραμμα σας, θα έχετε *numReducers* ενταμιευτές για μεταφορά ζευγαριών προς τα νήματα reducers. Τα νήματα mappers και κάθε νήμα reducer έχουν σχέση παραγωγού-καταναλωτή, και έτσι, στην υλοποίηση, οι προσβάσεις τους στον κοινό ενταμιευτή (του κάθε νήματος reducer) θα πρέπει να συγχρονίζονται. Συγκεκριμένα, τα νήματα mappers πρέπει να μπλοκάρονται και να περιμένουν όταν ο ενταμιευτής είναι γεμάτος ενώ το νήμα reducer πρέπει να περιμένει όταν ο ενταμιευτής είναι άδειος.

Σε αυτήν την εργασία, θα πρέπει να χρησιμοποιήσετε μεταβλητές συνθήκης στην υλοποίηση σας. **Αν η υλοποίηση σας κάνει οτιδήποτε busy-waiting, θα υπάρξει σημαντική ποινή.**

Η διαχείριση σφαλμάτων που θα πρέπει να κάνετε είναι, όταν συμβεί κάποιο σφάλμα, να τερματίζονται όλα τα νήματα και να διαγράφονται οι ενταμιευτές που χρησιμοποιήθηκαν για επικοινωνία και προσωρινή αποθήκευση δεδομένων, κλπ. Το ίδιο συμβαίνει και αν ο χρήστης διακόψει το πρόγραμμα με Control+C για παράδειγμα. Αν συμβεί σφάλμα το πρόγραμμα να ενημερώσει τον χρήστη με τον κωδικό λάθους του συγκεκριμένου σφάλματος.

Στην υλοποίηση σας πρέπει να χρησιμοποιήσετε system calls. Κατά συνέπεια αποκλείονται οι χρήσεις των fopen, fread, fwrite.

Επίσης τα αρχεία εισόδου θα πρέπει να διαμοιράζονται μεταξύ των mappers, έτσι ώστε να μην επεξεργαστεί ο ένας mapper 5 αρχεία ενώ ο άλλος 1 αρχείο.

Το αρχικό νήμα της main συνάρτησης του προγράμματος σας που δημιουργεί όλα τα νήματα mappers και reducers πρέπει να περιμένει την ολοκλήρωση των νημάτων ώστε να αποδεσμεύονται οι πόροι που χρησιμοποιούν.

Η εργασία θα γραφτεί σε γλώσσα C. Μπορείτε να γράψετε και C++, ωστόσο τα specifications της άσκησης έχουν γραφτεί για C (πχ η συνάρτηση ταξινόμησης). Σε κάθε περίπτωση θα πρέπει η άσκηση να συνοδεύεται με ένα Makefile που να αναλαμβάνει το «χτίσιμο» του προγράμματος σας.

Φροντίστε να έχετε ευανάγνωστο κώδικα και επεξηγηματικά σχόλια όπου χρειάζονται. Η άσκηση πρέπει να συνοδεύεται με ένα README που να περιγράφει περιληπτικά τι χρησιμοποιήθηκε στην άσκηση, τι υλοποιήθηκε και τι όχι και να εξηγεί τη λογική του προγράμματος σας.

Άσκηση 2: Networked mapper and reducer programs (50 points). Γράψτε ένα πρόγραμμα mapper και ένα πρόγραμμα reducer που υλοποιούν τις λειτουργίες των mapper/reducer και επικοινωνούν με πιθανώς πολλαπλά reducer/mapper προγράμματα μέσω δικτύου. Το πρόγραμμα mapper θα το τρέχετε από τη γραμμή εντολής με τα ακόλουθα ορίσματα

```
prompt> mapper [config-file] [numMappers] [input_folder] [mapper_ID]
```

Συγκεκριμένα τα ορίσματα είναι:

- **config-file:** ένα αρχείο που περιέχει πληροφορίες για τα reducers στα οποία θα στέλνει ζευγάρια το mapper. Κάθε γραμμή περιγράφει ένα reducer και έχει μορφή *reducer_id IP_addr port_number* όπου *reducer_id* είναι η ταυτότητα του reducer (0..numReducers-1), *IP_addr* είναι η διεύθυνση στην οποία τρέχει το συγκεκριμένο reducer και *port_number* ο αριθμός θύρας που άκουει το reducer για να δεχθεί ζευγάρια
- **numMappers:** ο αριθμός mappers που τρέχουν σε αυτό το MapReduce workload

- **input_folder:** το όνομα του κατάλογου που περιέχει τα αρχεία εισόδου του mapper
- **mapper_ID:** η ταυτότητα του mapper [0...numMappers-1]

Το πρόγραμμα reducer θα το τρέχετε από τη γραμμή εντολής με τα ακόλουθα ορίσματα

```
prompt> reducer [portNumber] [numMappers] [output_folder]
```

Συγκεκριμένα τα ορίσματα είναι:

- **portNumber:** ο αριθμός θύρας όπου θα άκουει το reducer για να δεχθεί ζευγάρια από τους mappers
- **numMappers:** ο αριθμός mappers από τα οποία θα περιμένει το reducer να δεχθεί ζευγάρια
- **output_folder:** το όνομα του κατάλογου όπου θα τοποθετείται το αρχείο εξόδου του reducer

Τα προγράμματα mappers και reducers θα συμπεριφέρονται όπως και οι αντίστοιχες διεργασίες που δημιουργήσατε στην δεύτερη εργασία με διαφορά στην επικοινωνία μεταξύ τους. Αντί για named pipes, η επικοινωνία θα γίνεται μεσω δικτυακών υποδοχών (network sockets). Θα πρέπει να σχεδιάσετε ένα πρωτόκολλο εφαρμογής για τη μεταφορά ζευγαριών ανάμεσα στο mapper και τα reducer με τα οποία επικοινωνεί. Στο πρωτόκολλο σας θα πρέπει να υπάρχει τρόπος να καταλάβει το κάθε reducer πότε του έχει στείλει το τελευταίο ζευγάρι ένα mapper. Στο README αρχείο θα πρέπει να περιγράψετε το πρωτόκολλο σας (είδος, σύνταξη, και σημασιολογία μηνυμάτων που ανταλλάζονται και κανόνες για την ανταλλαγή μηνυμάτων). Επίσης θα πρέπει να επιλέξετε ανάμεσα στο TCP και UDP για τη μεταφορά δεδομένων και να εξηγήσετε την επιλογή σας στο README. Αν επιλέξετε TCP θα πρέπει να υλοποιήσετε το reducer πρόγραμμα με έναν τρόπο μη σειριακό (π.χ., με ένα νήμα ανά σύνδεση με mapper, με ένα νήμα που εξυπηρετεί πολλαπλά sockets με τη χρήση της select system call, κλπ,) και να εξηγήσετε την επιλογή σας στο README. Αν από την άλλη, επιλέξετε UDP, θα πρέπει να σχεδιάσετε το πρωτόκολλο σας ώστε να είναι αξιόπιστη η μεταφορά – δηλαδή να εγγυάται ότι όλα τα ζευγάρια φτάνουν σε όλα τα reducers.

Στην υλοποίηση σας πρέπει να χρησιμοποιήσετε system calls. Κατά συνέπεια αποκλείονται οι χρήσεις των fopen, fread, fwrite.

Επίσης τα αρχεία εισόδου θα πρέπει να διαμοιράζονται μεταξύ των mappers, έτσι ώστε να μην επεξεργαστεί ο ένας mapper 5 αρχεία ενώ ο άλλος 1 αρχείο. Για αυτόν τον λόγο το mapper πρόγραμμα παίρνει τον αριθμό mappers και την ταυτότητα του mapper ώστε να επιλέγει τα κατάλληλα αρχεία εισόδου να διαβάσει.

Αν χρησιμοποιήσετε νήματα στα προγράμματα σας, το αρχικό νήμα της main συνάρτησης του προγράμματος σας που δημιουργεί τα αλλά νήματα πρέπει να περιμένει την ολοκλήρωση των νημάτων ώστε να αποδεσμεύονται οι πόροι που χρησιμοποιούν.

Η εργασία θα γραφτεί σε γλώσσα C. Μπορείτε να γράψετε και C++, ωστόσο τα specifications της άσκησης έχουν γραφτεί για C (πχ η συνάρτηση ταξινόμησης). Σε κάθε περίπτωση θα

πρέπει η άσκηση να συνοδεύεται με ένα Makefile που να αναλαμβάνει το «χτίσιμο» του προγράμματος σας.

Τέλος, φροντίστε να έχετε ευανάγνωστο κώδικα και επεξηγηματικά σχόλια όπου χρειάζονται.