ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

22 Φεβρουαρίου 2017

ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

Εξετάσεις Κανονικής Περιόδου Ακ. Έτους 2016-2017

Η εξέταση γίνεται με κλειστά βιβλία και σημειώσεις. Μπορείτε να έχετε μαζί σας μόνο μία κόλλα Α4. Διάρκεια εξέτασης $2^{3/4}$ ώρες.

Θέμα 1ο (60%):

Ο αλγόριθμος K-means κατανέμει *n* σημεία σε *k* συστάδες (clusters) ανάλογα με την «απόστασή» τους από το κέντρο της συστάδας. Ο αλγόριθμος λειτουργεί ως εξής:

Βήμα 1: Αρχικοποίηση του κέντρου κάθε συστάδας σε μία τυχαία αρχική τιμή
Βήμα 2: Ανάθεση κάθε σημείου στη εγγύτερη συστάδα
Βήμα 3: Ενημέρωση του κέντρου της συστάδας
Βήμα 4: Αν δεν άλλαξε κανένα σημείο συστάδα, τέλος, αλλιώς πήγαινε στο Βήμα 2

Θεωρήστε ότι κάθε σημείο βρίσκεται στο δισδιάστατο επίπεδο και έχει συντεταγμένες (x,y), οπότε η απόσταση δύο σημείων $\mathbf{v_1} = (x_1, y_1)$ και $\mathbf{v_2} = (x_2,y_2)$ είναι η ευκλείδεια απόσταση $\text{sqrt}((x_1-x_2)^2 + (y_1-y_2)^2)$, ενώ το κέντρο **c** μιας συστάδας *m* σημείων $\mathbf{v_i}$ (i = 1…m) δίνεται από τη σχέση $\mathbf{c} = \Sigma v_i / m$ (i = 1…m).

Α. Δώστε σειριακό ψευδοκώδικα που υλοποιεί τον παραπάνω αλγόρίθμο φροντίζοντας να είναι εμφανείς οι δομές δεδομένων που χρησιμοποιείτε για την αναπαράσταση των σημείων, των αποστάσεων των σημείων από τα κέντρα των συστάδων, των κέντρων των συστάδων και των συνόλων που περιγράφουν τις συστάδες. (7%)

Β. Καλείστε να σχεδιάσετε και να υλοποιήσετε παράλληλο πρόγραμμα για την εκτέλεση του παραπάνω αλγορίθμου σε υπολογιστική πλατφόρμα με πολλαπλούς επεξεργαστές που υποστηρίζει το μοντέλο κοινού χώρου διευθύνσεων.

I. Δώστε το γράφο των εξαρτήσεων αναδεικνύοντας το μέγιστο δυνατό παραλληλισμό. Επισημάνετε σε κάθε κόμβο του γράφου τον υπολογισμό που εκτελείται και σχολιάστε πιθανές διαφορές στο υπολογιστικό φορτίο ανάμεσα στις εργασίες. Σε περίπτωση που κάποιες εργασίες περιλαμβάνουν race condition επισημάνετέ το στο γράφο (π.χ. βάζοντας πιο έντονο περίγραμμα). (7%)
II. Δώστε έκφραση για τη μέγιστη δυνατή επιτάχυνση. (4%)
III. Υλοποιείστε παράλληλη έκδοση χρησιμοποιώντας *parallel for.* (4%)
IV. Σχεδιάστε το γράφο που εκτελείται από την υλοποίηση του παραπάνω ερωτήματος και σχολιάστε αν υπάρχουν διαφορές σε σχέση με το γράφο που σχεδιάσατε στο ερώτημα Β.Ι. (5%)
V. Υλοποιείστε παράλληλη έκδοση χρησιμοποιώντας *fork – join*. (5%)
VI. Σχεδιάστε το γράφο που εκτελείται από την υλοποίηση του παραπάνω ερωτήματος και σχολιάστε αν υπάρχουν διαφορές σε σχέση με το γράφο που σχεδιάσατε στο ερώτημα Β.Ι. (5%)

VII. Εντοπίστε στρατηγική υλοποίησης για το συγκεκριμένο πρόβλημα που αποφεύγει τα race conditions. Σχολιάστε το tradeoff ανάμεσα στο κόστος συγχρονισμού και στην ανισοκατανομή φορτίου και προτείνετε σχήμα που να βρίσκει συμβιβαστική λύση. (5%)

Γ. Αντίστοιχα με το ερώτημα Β, καλείστε να σχεδιάσετε και να υλοποιήσετε παράλληλο πρόγραμμα για σύστημα που υποστηρίζει προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων.

I. Δώστε κατανομή των δεδομένων για αυτό το μοντέλο (5%)
II. Δώστε ψευδοκώδικα υλοποίησης στο μοντέλο της ανταλλαγής μηνυμάτων. (13%)

Θέμα 2ο (25%):

Το *key-value pair* είναι ένας πολύ διαδεδομένος τρόπος αναπαράστασης δεδομένων. Πρόκειται για ένα σύνολο από δύο συνδεδεμένα στοιχεία: ένα *κλειδί (key)* το οποίο αποτελεί μοναδικό αναγνωριστικό για κάποια στοιχεία δεδομένων και μία *τιμή (value)* η οποία είναι είτε τα ίδια τα δεδομένα, είτε ένας δείκτης στην τοποθεσία των δεδομένων. Μία διαδεδομένη key-value pair δομή δεδομένων είναι το δυαδικό δέντρο αναζήτησης (*binary search tree*). Οι κόμβοι του δέντρου έχουν δύο διακριτά υποδέντρα (*αριστερό* και *δεξί*). Σε αυτή τη δομή τα κλειδιά βρίσκονται σε ταξινομημένη σειρά σύμφωνα με την ιδιότητα: το κλειδί κάθε κόμβου είναι μεγαλύτερο από οποιοδήποτε κλειδί στο αριστερό του υποδέντρο και μικρότερο από οποιοδήποτε κλειδί στο δεξί του υποδέντρο. Αυτή η μορφή επιτρέπει τη γρήγορη αναζήτηση (lookup), εισαγωγή (insertion) και διαγραφή (deletion) στοιχείων.

Παρακάτω σας δίνονται τρεις βασικές λειτουργίες σε ένα δυαδικό δέντρο αναζήτησης: *lookup* (αναζήτηση κλειδιού), *insert* (εισαγωγή κλειδιού) και *findMin* (εύρεση ελαχίστου κλειδιού).

I. Τροποποιήστε κατάλληλα τις λειτουργίες αυτές ώστε να επιτρέπεται ταυτόχρονη πρόσβαση στη δομή δεδομένων από πολλαπλά νήματα. Αντιμετωπίστε την ανάγκη συγχρονισμού με:

   a. coarse-grain locking (2%)
   b. fine-grain locking (7%)
   c. transactional memory (2%)

   Δώστε ψευδοκώδικα για κάθε μία από τις παραπάνω περιπτώσεις.

II. Αξιολογήστε τις τρεις παραπάνω στρατηγικές ως προς την επίδοση, την προγραμματιστική ευκολία και τη στιβαρότητα (robustness). (4%)

III. Σχολιάστε τη συμπεριφορά των παραπάνω στρατηγικών στις περιπτώσεις μικρού και μεγάλου δυαδικού δέντρου αναζήτησης. (3%)

IV. Ανάλογα με την εισαγωγή/διαγραφή κόμβων, ένα δυαδικό δέντρο μπορεί να εκφυλιστεί σε απλά συνδεδεμένη (ταξινομημένη) λίστα. Για παράδειγμα, όταν το δέντρο αρχικά είναι άδειο και γίνονται μόνο εισαγωγές κλειδιών που είναι απόλυτα ταξινομημένα μεταξύ τους, τότε προκύπτει μία ταξινομημένη λίστα. Τι αλλάζει σχετικά με το συγχρονισμό σε περίπτωση που το δυαδικό δέντρο αναζήτησης εκφυλιστεί σε απλή λίστα; (4%)

V. Προκειμένου να μην υπάρξει εκφυλισμός σε λίστα, εφαρμόζονται συνήθως τεχνικές ισοζυγισμού (rebalancing) στα δυαδικά δέντρα αναζήτησης, (π.χ. περιστροφές στους κόμβους του δέντρου είτε αριστερά είτε δεξιά). Έτσι προκύπτουν ισοζυγισμένα δέντρα όπως το AVL δέντρο αναζήτησης. Στη λειτουργία εισαγωγής κόμβου-κλειδιού σε AVL δέντρο, η εισαγωγή πραγματοποιείται αρχικά όπως στο απλό δυαδικό δέντρο αναζήτησης και έπειτα εκτελούνται περιστροφές στο δέντρο προς τα πάνω (κατεύθυνση από το σημείο που εισάχθηκε ο κόμβος προς τη ρίζα) προκειμένου να ισοζυγιστεί το δέντρο. Τί κίνδυνος εμφανίζεται σε αυτήν την περίπτωση; Ποιες στρατηγικές συγχρονισμού από τις παραπάνω θα μπορούσαν να εφαρμοστούν σε ένα τέτοιο δέντρο; (3%)

```
findMin (T)
{
   if (T.root == NULL)
      return SOME_SPECIAL_VALUE;

   curr = T.root;
   while (curr.leftChild != NULL)
      curr = curr.leftChild;

   return curr.key;
}
```

```
lookup (T, key)
{
   if (T.root == NULL)
      return 0;

   curr = T.root;

   while (curr != NULL) {
      if (curr.key == key)
         return 1;
      else if (curr.key > key)
         curr = curr.leftChild;
      else
         curr = curr.rightChild;
   }

   return 0;
}
```

```
insert (T, u)
{
   if (T.root == NULL) {
      T.root = u;
      return 1;
   }

   curr = NULL;
   next = T.root;

   while (next != NULL) {
      curr = next;

      if (curr.key == key)
         return 0;
      else if (curr.key > key)
         next = curr.leftChild;
      else
         next = curr.rightChild;
   }

   if (curr.key > key)
      curr.leftChild = u;
   else
      curr.rightChild = u;

   return 1;
}
```

Θέμα 3ο (20%):

Στο τελευταίο Top500 (Νοέμβριος 2016) εμφανίστηκαν αρκετά συστήματα με το νέο δίκτυο διασύνδεσης OmniPath. Το υπερυπολογιστικό κέντρο στο οποίο εργάζεστε αποφάσισε να κατασκευάσει τον επόμενο υπερυπολογιστή του με OmniPath. Σας ζήτησε να αξιολογήσετε διάφορες τοπολογίες για τη διασύνδεση των 10.000 κόμβων του νέου συστήματος με διακόπτες OmniPath 48 θυρών (48-port).

Α. Υπολογίστε το αριθμό των διακοπτών και σχεδιάστε ενδεικτικά τη διασύνδεση των κόμβων i) σε τοπολογία 5Δ-τόρου, ii) σε τοπολογία υπερκύβου. (10%)

Λάβετε υπόψη τα παρακάτω:

- Μία θύρα του διακόπτη μπορεί να συνδέεται είτε με κόμβο είτε με άλλο διακόπτη.
- Μία σύνδεση μεταξύ κόμβου/διακόπτη ή διακόπτη/διακόπτη είναι αμφίδρομη.
- Σε έναν διακόπτη συνδέονται περισσότεροι από ένας κόμβοι.

Β. Συγκρίνετε τις δύο τοπολογίες ως προς i) το κόστος (πλήθος διακοπτών και συνδέσεων) ii) την επίδοση (διάμετρος και εύρος ζώνης). Ποια τοπολογία είναι καταλληλότερη για το συγκεκριμένο σύστημα; (10%)