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

16 Φεβρουαρίου 2016

ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

Εξετάσεις Κανονικής Περιόδου Ακ. Έτους 2015-2016

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 (50%):

Δίνεται ο παρακάτω υπολογιστικός πυρήνας που αποτελεί μέρος της επίλυσης γραμμικού συστήματος:

```
for (k = 0; k < N; k++)
   for (i = k+1; i < N; i++) {
      l = A[i][k] / A[k][k];
      for (j = k+1; j < N; j++)
         A[i][j] = A[i][j] – l*A[k][j];
   }
```

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

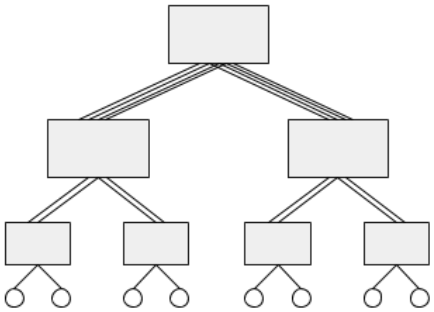
I. Περιγράψτε πως διαφοροποιούνται η task centric και data centric τακτικές σχεδιασμού στο συγκεκριμένο πρόβλημα. (3%)
II. Ακολουθήστε την task centric λογική, περιγράψτε τις εργασίες (tasks) που ορίσατε και χαρακτηρίστε τα δεδομένα κάθε εργασίας (shared, distributed, replicated). (4%)
III. Σχεδιάστε το γράφο των εξαρτήσεων για N = 5. (6%)
IV. Επισημάνετε το μέγιστο μονοπάτι και υπολογίστε τη μέγιστη επιτάχυνση για N=5. Γενικεύστε τον υπολογισμό της μέγιστης επιτάχυνσης για οποιοδήποτε N. (4%)
V. Εντοπίστε τα σημεία στο γράφο όπου απαιτείται κατάλληλος συγχρονισμός για την ορθή εκτέλεση και περιγράψτε με ποιο βασικό μηχανισμό μπορεί να υλοποιηθεί ο συγχρονισμός αυτός, π.χ. με locks, condition variables, barriers ή με σχήματα που μπορείτε να προτείνετε εσείς. (4%)
VI. Έστω ότι επιλέγετε στατική απεικόνιση των εργασιών σε επεξεργαστικές οντότητες, ομαδοποιώντας τις εργασίες κάθε γραμμής του πίνακα στην ίδια οντότητα εκτέλεσης. Θα επιλέξετε σειριακή (sequential) ή κυκλική (cyclic) απεικόνιση; Δικαιολογήστε την απάντησή σας. (4%)
VII. Περνώντας στο στάδιο της υλοποίησης, εντοπίστε τους βρόχους που είναι παράλληλοι και δώστε ψευδοκώδικα που να τους παραλληλοποιεί. (4%)
VIII. Δώστε υλοποίηση στο μοντέλο fork-join. (5%)
IX. Με τις υλοποιήσεις των δύο προηγούμενων ερωτημάτων πετύχατε την εκτέλεση όπως περιγράφεται στο γράφο των εξαρτήσεων που δώσατε στο ερώτημα II; Αν όχι, με ποιον τρόπο μπορείτε να την πετύχετε; (3%)

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

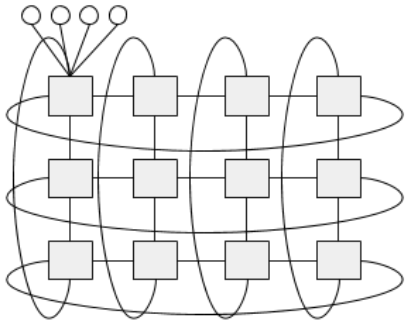
I. Τι θα πρέπει να τροποποιήστε στο σχεδιασμό σας σε σχέση με την περίπτωση Α; (3%)
II. Δώστε ψευδοκώδικα υλοποίησης στο μοντέλο της ανταλλαγής μηνυμάτων. (10%)

Θέμα 2ο (20%):

Στο Σχήμα 1 δίνεται η τυπική τοπολογία fat tree για τη διασύνδεση 8 κόμβων (κύκλοι). Παρατηρήστε ότι σε υψηλότερα επίπεδα του fat tree απαιτείται η χρήση μεγαλύτερων switches (τετράγωνα) και ότι το επίπεδο που βρίσκεται στη ρίζα χρειάζεται ίδιου μεγέθους switch με το αμέσως χαμηλότερο. Παρατηρήστε επίσης τη βασική ιδιότητα ενός fat tree: στα ενδιάμεσα επίπεδα, για κάθε switch, ο αριθμός των συνδέσμων προς τα κάτω ισούται με τον αριθμό των συνδέσμων προς τα επάνω (uplinks = downlinks).



Σχήμα 1: fat tree



Σχήμα 2: 2Δ τόρος

I. Ο ελληνικός υπερυπολογιστής ARIS χρησιμοποιεί 36-port switches (τεχνολογίας Infiniband) για τη διασύνδεση 648 κόμβων σε τοπολογία fat tree δύο επιπέδων. Πόσα switches χρησιμοποιούνται στο ARIS; Δώστε ενδεικτικό σχήμα με τη συνδεσμολογία τους. (7%)

II. Μια εναλλακτική τοπολογία που χρησιμοποιείται ευρέως σε υπερυπολογιστές είναι αυτή του τρισδιάστατου τόρου (3Δ-τόρος). Το Σχήμα 2 απεικονίζει έναν 2Δ-τόρο. Παρατηρήστε ότι τα switches διασυνδέονται στην τοπολογία του τόρου, ενώ περισσότεροι από ένας κόμβοι μπορούν να συνδεθούν πάνω σε ένα switch. Χρησιμοποιώντας 36-port switches σαν αυτά του ARIS, περιγράψτε πώς θα συνδεθούν οι 648 κόμβοι του ARIS σε τοπολογία 3Δ-τόρου, με χρήση διπλών συνδέσμων σε κάθε κατεύθυνση. Συγκρίνετε τη νέα τοπολογία με αυτή του ARIS ως προς το *κόστος* και την *επίδοση*. Δίνεται ότι το εύρος τομής του fat tree είναι N και του 3Δ-τόρου είναι $2N^{2/3}$, όπου N το πλήθος των κόμβων του δικτύου. (7%)

III. Τι αλλαγές πρέπει να γίνουν στα δίκτυα των δύο προηγούμενων ερωτημάτων, προκειμένου να διπλασιαστούν οι κόμβοι του συστήματος; Συγκρίνετε τις τοπολογίες ως προς την *κλιμακωσιμότητά* τους. (6%)

Θέμα 3ο (30%):

Στη συνέχεια δίνεται σε ψευδοκώδικα ο αλγόριθμος του Dijkstra που βρίσκει τα ελάχιστα μονοπάτια από την πηγή *s* προς όλους τους κόμβους ενός γράφου *G(V, E)*. Υπενθυμίζεται ότι ο αλγόριθμος λειτουργεί επαναληπτικά, ενημερώνοντας την απόσταση κάθε κόμβου *v* από την πηγή *d[v],* και τον πρόγονό του *π[v]*. Ο πίνακας *w(u,v)* περιέχει το βάρος των ακμών του γράφου. Ο αλγόριθμος χρησιμοποιεί ουρά προτεραιότητας *Q* η οποία περιλαμβάνει τις λειτουργίες *Insert*, (εισαγωγή κόμβου στην ουρά), *ExtractMin* (εξαγωγή του κόμβου με τη μικρότερη απόσταση) και *DecreaseKey* (ενημέρωση της θέσης του στοιχείου στην ουρά μετά από αλλαγή της απόστασής του). Η ουρά είναι υλοποιημένη σαν ταξινομημένη απλά συνδεδεμένη λίστα, με δύο κόμβους φρουρούς (*head, tail*) και κλειδί (*key*) την απόσταση του κόμβου από την πηγή. Ο ψευδοκώδικας για τις λειτουργίες δίνεται μετά τα ζητούμενα.

```
foreach node u do                  // Αρχικοποίηση της ουράς
  d[u] = INF;
  π[u] = -1;
  newNode = newNode(u, INF);
  Insert(Q, newNode);
end

DecreaseKey(Q, v, d[s], 0);        // Αρχικοποίηση του κόμβου πηγής

while Q is not empty do
  u = ExtractMin(Q);               // Εξαγωγή κόμβου u με τη μικρότερη απόσταση
  foreach v adjacent to u do       // Για κάθε γείτονα
    sum = d[u] + w(u,v);              // υπολογίζεται η απόσταση από τον u
    if (sum < d[v]) then              // αν η νέα απόσταση είναι μικρότερη
      d[v] = sum;                        // ενημερώνεται η απόσταση
      π[v] = u;                          // ενημερώνεται η πρόγονος
      DecreaseKey(Q, v, d[v], sum);  // ενημερώνεται η θέση στην ουρά
  end
end
```

I. Θέλουμε να εκτελέσουμε τον παραπάνω αλγόριθμο σε σύστημα με πολλαπλούς επεξεργαστές που υποστηρίζει προγραμματιστικό μοντέλο κοινού χώρου διευθύνσεων. Χρησιμοποιήστε το μοντέλο fork-join και επισημάνετε ποια tasks μπορούν να εκτελεστούν παράλληλα. (5%)

II. Αντιμετωπίστε την ανάγκη συγχρονισμού σε κοινά δεδομένα:
   a. με coarse-grain locking (5%)
   b. με fine-grain locking (10%)
   c. με τη χρήση transactional memory (5%)

   Δώστε ψευδοκώδικα για κάθε μία από τις παραπάνω περιπτώσεις

III. Σχολιάστε τις τρεις παραπάνω στρατηγικές ως προς την *επίδοση*, την *προγραμματιστική ευκολία* και τη *στιβαρότητα* (robustness). (5%)

```
Insert(Q, u)
  pred = Q.head;
  curr = pred.next;

  while (curr.key < u.key)
    pred = curr;
    curr = curr.next;
  end

  if curr.key == u.key then
    return 0;

  u.next = curr;
  pred.next = u;
  return 1;
```

```
ExtractMin(Q)
  ret = Q.head.next;
  if ret == Q.tail then
    return NULL;
  else
    return ret;
```

```
DecreaseKey(Q, u, oldKey, newKey)
  pred = Q.head;
  curr = pred.next;

  while (curr.key < newKey)
    pred = curr;
    curr = curr.next;
  end

  newKey_pred = pred;

  while (curr.key < oldKey)
    pred = curr;
    curr = curr.next;
  end

  pred.next = curr.next;
  curr.next = newKey_pred.next;
  newKey_pred.next = curr;
```