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

ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ
9ο εξάμηνο ΗΜΜΥ, ακαδημαϊκό έτος 2004-05

1 Γενικά

Σκοπός της εργαστηριακής άσκησης του μαθήματος είναι η σχεδίαση και υλοποίηση παράλληλου αλγορίθμου, που να εφαρμόζει τη μέθοδο Conjugate Gradient (CG) για την επίλυση γραμμικού συστήματος

$$Az = x$$

όπου A αραιός πίνακας συντελεστών διαστάσεων $N \times N$, x ο πίνακας-στήλη των σταθερών όρων του συστήματος και z ο πίνακας των αγνώστων. Η μέθοδος επίλυσης CG βασίζεται στον ακόλουθο αλγόριθμο:

---

1 $z = 0$;
2 $z_0 = 0$;
3 $p = r = x$;
4 $\rho = r^T r$;
5 **repeat**
6 $\quad q = Ap$;
7 $\quad \alpha = \frac{\rho}{p^T q}$;
8 $\quad z_0 = z$;
9 $\quad z = z + \alpha p$;
10 $\quad \rho_0 = \rho$;
11 $\quad r = r - \alpha q$;
12 $\quad \rho = r^T r$;
13 $\quad \beta = \frac{\rho}{\rho_0}$;
14 $\quad p = r + \beta p$;
15 **until** $|z - z_0| < eps|z_0|$;

Στον παραπάνω αλγόριθμο, οι βοηθητικοί πίνακες z_0, r, p, q είναι διάστασης $N \times 1$, ενώ οι βαθμωτές μεταβλητές (scalars) συμβολίζονται με ελληνικά γράμματα ($\alpha, \beta, \rho, \rho_0$). Ο αλγόριθμος ελέγχει τη σύγκλιση ως προς κάποια δεδομένη ακρίβεια eps, που καθορίζεται από το χρήστη. Η λύση του αλγορίθμου παρέχεται στον πίνακα στήλη z, σε περίπτωση που επιτευχθεί η σύγκλιση. Ο αλγόριθμος εκτελείται για μέγιστο πλήθος επαναλήψεων ίσο με num_steps, που επίσης καθορίζεται από το χρήστη. Σε περίπτωση που η σύγκλιση δεν έχει επιτευχθεί σε num_steps βήματα, ο αλγόριθμος ολοκληρώνεται και ο χρήστης ενημερώνεται σχετικά.

Η μέθοδος CG χρησιμοποιείται στα NAS Parallel Benchmarks (NPB), που αποτελούν ευρύτατα διαδεδομένα μετροπρογράμματα για την αξιολόγηση επίδοσης παράλληλων αρχιτεκτονικών και προγραμματιστικών μοντέλων. Περισσότερες πληροφορίες τόσο για τα NAS Parallel Benchmarks, όσο και για τον CG ειδικότερα, παρέχονται στο [1].

2 Ζητούμενα

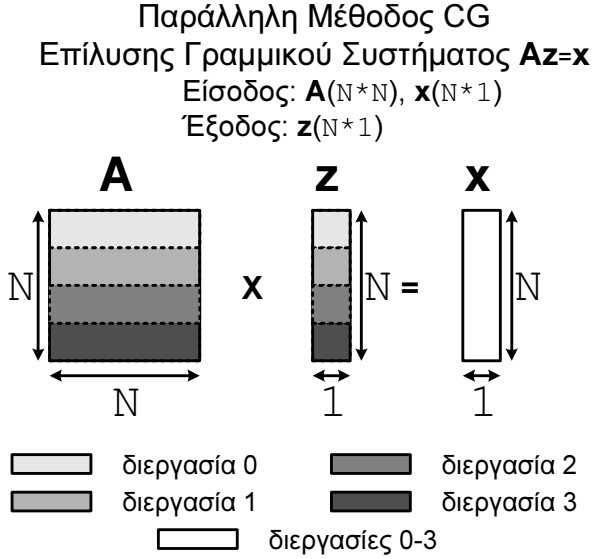
Ζητούμενο της εργαστηριακής άσκησης είναι η υλοποίηση παράλληλου προγράμματος σε MPI, καθώς και προαιρετικά υβριδικού προγράμματος με χρήση MPI+OpenMP, που να υλοποιούν τον αλγόριθμο της ενότητας 1. Αφετηριακό σημείο για τη σχεδίαση του παράλληλου προγράμματος αποτελεί η υιοθέτηση στρατηγικής *κατανομής των υπολογισμών* (computation distribution) στις επιμέρους διεργασίες. Οι δυνατότητες εναλλακτικών σχημάτων κατανομής υπολογισμών προκύπτουν από τη μελέτη της φύσης του αλγορίθμου, ενώ η επιλογή ενός συγκεκριμένου σχήματος θα καθορίσει τελικά και τις αντίστοιχες απαιτήσεις *κατανομής των δεδομένων* (data distribution) μεταξύ των διεργασιών. Ιδιαίτερα για τον αλγόριθμο CG, προτεραιότητα στην παραλληλοποίηση θα πρέπει να δοθεί στον υπολογισμό της γραμμής 6, καθώς απαιτεί N^2 πολλαπλασιασμούς και συνεπώς αναμένεται να επιβαρύνει καθοριστικά το συνολικό χρόνο εκτέλεσης. Ιδιαίτερη έμφαση πρέπει να δοθεί επίσης στις γραμμές 9, 11 και 14, καθώς στην πρώτη ουσιαστικά γίνεται η ενημέρωση της τιμής του αγνώστου διανύσματος z, ενώ στις άλλες δύο ενημερώνονται τα βοηθητικά διανύσματα r και p, αντίστοιχα.

Στο σχήμα 1 προτείνεται μια δυνατή κατανομή των βασικών δεδομένων του προβλήματος σε διεργασίες. Έτσι, ο πίνακας των συντελεστών A διαμερίζεται σε υποπίνακες με οριζόντια κατάτμηση. Το ίδιο ισχύει και για τον πίνακα των αγνώστων z, έτσι ώστε κάθε διεργασία να υπολογίζει τελικά ένα μέρος της συνολικής λύσης του προβλήματος. Ο πίνακας των σταθερών όρων x γνωστοποιείται ολόκληρος σε όλες τις διεργασίες. Η συγκεκριμένη προσέγγιση του σχήματος 1 δεν είναι δεσμευτική, και παρέχεται απλά ως πρόταση. Οποιαδήποτε άλλη αποδοτική παραλληλοποίηση του αλγορίθμου CG είναι αποδεκτή.

Σε γενικές γραμμές, μεταξύ των διεργασιών απαιτείται επικοινωνία για δύο κυρίως λόγους:

1. Για τον έλεγχο της καθολικής σύγκλισης του αλγορίθμου. Το παράλληλο πρόγραμμα θα πρέπει να περατώνει τη λειτουργία του το πολύ σε max_steps βήματα, ή και ακόμα νωρίτερα, εφόσον έχουν συγκλίνει όλες οι διεργασίες.

2. Για την ανταλλαγή δεδομένων μεταξύ των διεργασιών.



Εικόνα 1: Παράδειγμα απεικόνισης βασικών πινάκων του αλγορίθμου σε 4 διεργασίες

Με βάση τα παραπάνω, ζητούνται από κάθε ομάδα τα εξής:

- Η **υλοποίηση παράλληλου προγράμματος** σε MPI ή προαιρετικά σε υβριδικό MPI+OpenMP, που να εφαρμόζει τον αλγόριθμο CG για την επίλυση γραμμικού συστήματος $Az = x$. Η υλοποίηση του υβριδικού προγράμματος είναι προαιρετική, και πριμοδοτείται με μία επιπλέον μονάδα. Για να ελέγξετε την ορθότητα του προγράμματός σας, σας προτείνουμε να χρησιμοποιήσετε απλές περιπτώσεις για τους A, z, όπως

$$A[i][j] = \begin{cases} 0 & , \quad i \neq j \\ i & , \quad i = j \end{cases} \tag{1}$$

$$x[i] = i \tag{2}$$

  Σε κάθε περίπτωση, η αρχικοποίηση των A, z να γίνεται με δομημένο τρόπο, ώστε να μπορεί εύκολα να εξεταστεί η επίλυση ενός άλλου συστήματος. Επίσης, το πρόγραμμα θα πρέπει να είναι παραμετρικό ως προς τον αριθμό των διεργασιών P, το πλήθος T των νημάτων (για υβριδικό), τη διάσταση N του προβλήματος, το σφάλμα eps καθώς και το μέγιστο αριθμό βημάτων max_steps. Η έξοδος του προγράμματος, σε περίπτωση που υπάρχει σύγκλιση, θα πρέπει να γράφεται σε αρχείο κειμένου, που θα έχει την ακόλουθη μορφή:

  $N\ eps$
  $z[0]$
  $z[1]$
  ...
  $z[N-1]$

  Σε περίπτωση που δεν επιτυγχάνεται η σύγκλιση υπό το δεδομένο αριθμό βημάτων

max_steps, θα πρέπει να ενημερώνεται ο χρήστης σχετικά. Όλα τα δεδομένα να είναι τύπου `float`.

- Η **πραγματοποίηση μετρήσεων** για μέγεθος $N = 3000, 3500, 4000$ και πλήθος επεξεργαστών $P = 2, 4, 8, 16$, και η σύγκριση των αντίστοιχων χρόνων εκτέλεσης με εκείνους του σειριακού προγράμματος. Συγκεκριμένα, για κάθε τιμή του N να παρασταθεί η *επιτάχυνση* που επιτυγχάνεται ως προς το σειριακό αλγόριθμο σαν συνάρτηση του πλήθους P των επεξεργαστών. Για τη μέτρηση των χρόνων προτείνεται να χρησιμοποιηθεί η συνάρτηση βιβλιοθήκης gettimeofday. Πιο συγκεκριμένα, να ακολουθηθεί η εξής διαδικασία:

```
/* ορισμός μεταβλητών, δεσμεύσεις μνήμης,
   αρχικοποιήσεις πινάκων, αρχικοποιήσεις MPI */
MPI_Barrier(MPI_COMM_WORLD);
gettimeofday(start,(struct timezone*)NULL);
/* παράλληλος αλγόριθμος CG */
...
MPI_Barrier(MPI_COMM_WORLD);
gettimeofday(finish,(struct timezone*)NULL);
duration=f(finish-start);
/* εκτύπωση εξόδου σε χρήστη, αποτελεσμάτων σε αρχείο */
```

  Παρατηρείστε ότι κατά την μέτρηση χρόνων ενδιαφέρει **μόνο** το υπολογιστικό κομμάτι του αλγορίθμου, και όχι η φάση αρχικοποίησης ή εκτύπωσης των αποτελεσμάτων. Ο συγχρονισμός των διεργασιών μέσω της `MPI_Barrier` πριν κάθε μέτρηση χρόνου θα οδηγήσει στον υπολογισμό του χρόνου από την (συγχρονισμένη) έναρξη όλων των διεργασιών μέχρι την περάτωση της τελευταίας χρονικά διεργασίας, που αποτελεί άλλωστε και τον πραγματικό χρόνο εκτέλεσης του παράλληλου προγράμματος.

- **Σύντομη αναφορά** με δικαιολόγηση των κυριότερων σχεδιαστικών επιλογών του προγράμματος, επεξήγηση των βασικών δομών και την γενικής λειτουργίας αυτού, παρουσίαση μετρήσεων με τα διαγράμματα επιτάχυνσης, καθώς και σχολιασμό των πειραματικών αποτελεσμάτων.

3 Διευκρινίσεις

Η ανάπτυξη της άσκησης θα γίνει στις συστοιχίες υπολογιστών του εργαστηρίου, τα 16 kids (kid1-kid16) για απλό MPI και τα 8 twins (twin1-twin4, twin6, twin7, twin9, twin12) σε περίπτωση υβριδικού προγράμματος. Για να συνδεθούμε στα μηχανήματα αυτά, ακολουθούμε τις οδηγίες που βρίσκονται στη σελίδα του μαθήματος (`http://www.cslab.ece.ntua.gr/courses/pps`).

Για ομοιομορφία των μετρήσεων, συνίσταται να χρησιμοποιηθεί ο Intel C++ compiler (icc) κατά τη μεταγλώττιση του σειριακού, καθώς και η εγκατάσταση MPI που βασίζεται στον icc για τη μεταγλώττιση των παραλλήλων προγραμμάτων. Το παραπάνω

μπορεί να επιτευχθεί με χρήση του `/usr/local/mpich-intel/bin/mpicc` για μεταγλώττιση σε όλες τις περιπτώσεις (χρειάζεται `-openmp` για το υβριδικό), ενώ το script `/usr/local/mpich-intel/bin/mpirun` μπορεί να χρησιμοποιηθεί για την εκτέλεση των παράλληλων προγραμμάτων. Επίσης, για ταχύτερους χρόνους εκτέλεσης, προτείνεται η χρήση επιπέδου βελτιστοποίησης `-O3` κατά τη μεταγλώττιση.

Έτσι, π.χ. για την μεταγλώττιση και εκτέλεση ενός προγράμματος MPI mcg.c, συνίσταται η εξής ακολουθία εντολών:

```
/usr/local/mpich-intel/bin/mpicc -o mcg mcg.c -O3
/usr/local/mpich-intel/bin/mpirun -np 16 mcg
```

Σε περίπτωση υβριδικού προγράμματος hcg.c, η πρώτη εντολή τροποποιείται ως εξής:

```
/usr/local/mpich-intel/bin/mpicc -o hcg hcg.c -O3 -openmp
```

Στο υβριδικό τέλος, επιθυμούμε σε κάθε SMP κόμβο της συστοιχίας των twins να εκκινείται μία μόνο διεργασία MPI, η οποία με τη σειρά της θα αξιοποιεί τους δύο επεξεργαστές του κόμβου με ισάριθμα νήματα OpenMP. Επειδή η εγκατάσταση του MPI θα εκκινεί εξ ορισμού δύο διεργασίες σε κάθε SMP κόμβο, απαιτείται η χρήση κατάλληλου machinefile. Έτσι, η εκτέλεση του παράλληλου προγράμματος στην περίπτωση αυτή θα πρέπει να γίνεται με

```
/usr/local/mpich-intel/bin/mpirun -np 8 -machinefile machines hcg
```

όπου το αρχείο machines θα περιέχει τα εξής:

```
twin1
twin2
twin3
twin4
twin6
twin7
twin9
twin12
```

Περισσότερες πληροφορίες σχετικά με το MPICH και τις SMP συστοιχίες υπάρχουν στο `http://www-unix.mcs.anl.gov/mpi/mpich/docs/mpichman-chp4/node14.htm#Node19`

Καλή επιτυχία!

Βιβλιογραφία

[1] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. RNR Technical Report RNR-94-007, NAS Systems Division, NASA Ames Research Center, Moffett Field, CA 94035-1000, March 1994.