ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΗΡΥ 418 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΠΑΡΑΛΛΗΛΩΝ ΚΑΙ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018-19

Άσκηση 1: Χρήση OpenMP και pthreads

Εαρινό εξάμηνο 2018-19
Ν. Αλαχιώτης

Περιγραφή

Στην άσκηση αυτή θα χρησιμοποιήσετε το OpenMP Application Protocol Interface (API) και ένα υποσύνολο των συναρτήσεων του POSIX threads standard (γνωστό και ως pthreads) για να επιταχύνετε τον αλγόριθμο Smith-Waterman όπως μπορεί να χρησιμοποιηθεί για τοπική ευθυγράμμιση ακολουθιών. Αρχικά μελετήστε και κατανοήστε πως λειτουργεί ο αλγόριθμος στην απλή περίπτωση εφαρμογής του με linear gap penalty.

Δημιουργία σειριακού προγράμματος (reference)

Δημιουργείστε ένα σειριακό πρόγραμμα το οποίο θα δέχεται ως είσοδο από το command line ένα σύνολο απο command-line **flags** και *arguments* ως εξής:

./project1 **-name** *ID* **-input** *PATH* **-match** *INT1* **-mismatch** *INT2* **-gap** *INT3*

Τα flags μπορούν να δίνονται από τον χρήστη σε τυχαία σειρά. Το ID είναι string και θα χρησιμοποιηθεί ως διακριτικό για την δημιουργία του ζητούμενου αρχείου εξόδου του προγράμματός σας. Το argument PATH δίνει το path σε ένα αρχείο εισόδου τύπου text, π.χ. /home/myusername/dataset.txt, το οποίο περιέχει έναν αριθμό D_SIZE από ζευγάρια ακολουθιών χαρακτήρων, με την κάθε ακολουθία να βρίσκεται σε ξεχωριστή γραμμή ή να εκτείνεται σε περισσότερες γραμμές για ευκολία ανάγνωσης. Η μορφή του αρχείου φαίνεται με ένα παράδειγμα παρακάτω:

dataset.txt

2

Q: abc
D: xxxabxcxxxaabbcc

Q: aaabcd
D: abababcababababcd

Θα σας δωθούν input αρχεία για testing. Τα επόμενα arguments, INT1, INT2 και INT3 προσδιορίζουν το scoring scheme του αλγορίθμου για MATCH, MISMATCH και GAP αντίστοιχα. Το πρόγραμμά σας θα πρέπει να εκτελεί τον αλγόριθμο για κάθε ζευγάρι ακολουθιών Q-D και να γράφει στο αρχείο εξόδου (Report_ID.txt) τις ακολουθίες Q και D και στη συνέχεια το αποτέλεσμα της ευθυγράμμισης με το καλύτερο score. Αν υπάρχουν περισσότερες από μία τοπικές ευθυγραμμίσεις με το ίδιο (μέγιστο) score θα πρέπει να εμφανίζονται όλες στο αρχείο εξόδου. Για κάθε τοπική ευθυγράμμιση θα πρέπει να αποθηκεύετε επίσης στο αρχείο εξόδου την αρχή και το τέλος (θέση του πρώτου και τελευταίου χαρακτήρα) στην ακολουθία D, το κοινό μέγεθος των δύο ευθυγραμμισμένων κομματιών των ακολουθιών, καθώς και το score του αλγορίθμου.

Για παράδειγμα για το πρώτο ζευγάρι ακολουθιών Q-D μπορεί να προκύψουν δύο τοπικές ευθυγραμμίσεις με το ίδιο score, οπότε το αποτέλεσμα θα είναι:

Q: abc
D: xxxabxcxxxaabbcc

Match 1 [Score: 8, Start: 3, Stop: 6]
D: a b x c
Q: a b – c

Match 2 [Score: 8, Start: 10, Stop: 14]
D: a a b b c
Q: a – b – c

Κατανοήστε πως προέκυψαν τα παραπάνω αποτελέσματα για το πρώτο ζευγάρι ακολουθιών Q-D στο dataset.txt. Τα scores που χρησιμοποιήθηκαν είναι: match = 3, mismatch = -1, gap = -1. Ο αλγόριθμος Smith-Waterman αποτελείται απο δύο διακριτά υπολογιστικά βήματα, τον υπολογισμό του scoring matrix (υπολογισμός πίνακα βαθμολόγησης) και το traceback (βήματα οπισθοδρόμησης). Για κάθε ζευγάρι ακολουθιών Q-D απαιτείται η δημιουργία ενός πίνακα βαθμολόγησης, ενώ μπορεί να χρειαστούν περισσότερα από ένα traceback steps, όπως στο παραπάνω παράδειγμα που χρειάστηκαν δύο: Match 1 και Match 2. Διαφορετικά ζευγάρια ακολουθιών απαιτούν διαφορετική ευθυγράμμιση η οποία μπορεί να απαιτεί gaps (–) και στην ακολουθία D. Στο παραπάνω παράδειγμα έτυχε να μην χρειάζονται gaps στο D για ευθυγράμμιση.

Το πρόγραμμα σας θα τυπώνει στην οθόνη μόνο στοιχεία ενδεικτικά της απόδοσης:

A) συνολικός αριθμός ζευγαριών ακολουθιών Q-D,
B) συνολικός αριθμός κελιών που πήραν τιμή,
C) συνολικός αριθμός απο traceback steps,
D) συνολικός χρόνος εκτέλεσης προγράμματος,
E) συνολικός χρόνος υπολογισμού κελιών,
F) συνολικός χρόνος για το traceback,
G) συνολικός αριθμός κελιών που παίρνουν τιμή στη μονάδα του χρόνου (CUPS: Cell Updates Per Second) βάση του συνολικού χρόνου εκτέλεσης, και
H) CUPS βάση του χρόνου υπολογισμού κελιών.

Ένας ενδεικτικός τρόπος μέτρησης του χρόνου στον κώδικά σας φαίνεται παρακάτω:

```
#include <sys/time.h>

double gettime(void)
{
  struct timeval ttime;
  gettimeofday(&ttime, NULL);
  return ttive.tv_sec+ttime.tv_usec * 0.000001;
}

double time0 = gettime();
code
double time1 = gettime();
```

elapsed time: time1-time0

Χρήση OpenMP και Pthreads

Στη συνέχεια μελετήστε τη συμπεριφορά του κώδικα σας για μεγάλα dataset sizes, και αφού εντοπίσετε τα σημεία που μπορούν και πρέπει να υπολογιστούν παράλληλα για λόγους απόδοσης, επιταχύνετε την εκτέλεσή του προγράμματος σας με την χρήση του OpenMP API. Υπάρχουν περισσότεροι απο ένας πιθανοί τρόποι παραλληλισμού, με διαφορετικό granularity και κατά συνέπεια διαφορετικό computation-to-communication ratio, τους οποίους πρέπει να υλοποιήσετε και να αξιολογήσετε (τουλάχιστον δύο). Για κάθε διαφορετικό τρόπο παραλληλοποίησης χρησιμοποιείστε conditional compilation directives στο ίδιο set απο C αρχεία.

Τροποποιήστε τον κώδικα σας ώστε να δέχετε απο τον χρήστη και τον αριθμό των threads που θα δημιουργηθούν μέσω του command-line flag **-threads** *INT4*. Το παράλληλο πρόγραμμά σας με OpenMP θα πρέπει να δημιουργεί το αρχείο εξόδου Report_*ID*_OMP_*INT4*.txt.

Σε Linux, για να κάνετε compile για OpenMP, χρησιμοποιείτε τον gcc ως εξής:

```
gcc -fopenmp -o <όνομα εκτελέσιμου> <όνομα αρχείου.c>
```

Στη συνέχεια αντικαταστήστε τον OpenMP κώδικα με pthreads και επαναλάβετε τις ίδιες μετρήσεις για όλες τις εναλλακτικές λύσεις παραλληλοποίησης. Το παράλληλο πρόγραμμά σας με Posix threads θα πρέπει να δημιουργεί το αρχείο εξόδου Report_*ID*_PTH_*INT4*.txt. Οι βασικές προσθήκες που θα πρέπει να γίνουν προκειμένου να εκτελέσετε ένα πρόγραμμα με pthreads είναι οι εξής:

1. Να προσθέσετε το header file #include <pthread.h> στο πρόγραμμα.
2. Για να τρέξετε το πρόγραμμα, χρησιμοποιήστε το switch –lpthread, για παράδειγμα:

```
gcc –lpthread <όνομα αρχείου.c> -o <όνομα εκτελέσιμου>
```

Για τις λύσεις με pthreads απαιτείται η χρήση του POSIX threads API της C, όχι higher level υλοποιήσεις που βασίζονται σε αυτό και υπάρχουν online.

Load Imbalance και Dataset size

Οι υλοποιήσεις σας θα πρέπει να λαμβάνουν υπόψη πιθανά θέματα load balancing και να υπάρχει κατάλληλος τρόπος αντιμετώπισης αν και όπου χρειάζεται.

Όλες οι υλοποιήσεις σας θα πρέπει να μπορούν να λειτουργήσουν σωστά χωρίς περιορισμό στο πλήθος των ζευγαριών Q-D, δεδομένου ότι ο μεγαλύτερος πίνακας βαθμολόγησης (ανα thread) για ένα dataset μπορεί να αποθηκευτεί στη μνήμη.

Παραδοτέα:

1. source code για όλες τις υλοποιήσεις: σειριάκη υλοποίηση, παράλληλες υλοποιήσεις με OpenMP, παράλληλες υλοποιήσεις με pthreads
2. run.sh script (executable χωρίς input parameters) που θα καλεί όλες τις υλοποιήσεις για ένα text αρχείο.
3. Αναφορά (PDF) που να περιγράφει και να σχολιάζει τις παράλληλες υλοποιήσεις και την λογική που ακολουθήσατε, να δείχνει τα αποτελέσματα σας σε σύγκριση με τον σειριακό κώδικα (χρόνοι εκτέλεσης, CUPS, speedups για διαφορετικό αριθμό απο threads), καθώς και συμπεράσματα και παρατηρήσεις για τη χρήση OpenMP και pthreads.

Υποβολή και Βαθμολόγηση

Ημερομηνία υποβολής: **27 Μαρτίου**

Για να θεωρηθεί ολοκληρωμένη η υποβολή του 1ου project και να βαθμολογηθεί θα πρέπει να υπάρχουν τουλάχιστον τα εξής:
a) source code (65%)
b) run script run.sh (executable) (5%)
c) Αναφορά σε PDF format (30%)

Το slide 3 της πρώτης διάλεξης αναφέρει τις απαιτήσεις λεπτομερώς.

Πληροφορίες για τον αλγόριθμο Smith-Waterman

https://en.wikipedia.org/wiki/Smith-Waterman_algorithm
https://en.wikipedia.org/wiki/Smith-Waterman_algorithm#Linear

Πληροφορίες για OpenMP

https://computing.llnl.gov/tutorials/openMP/
http://www.openmp.org

Πληροφορίες για POSIX threads

https://computing.llnl.gov/tutorials/pthreads/
http://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html
https://www.ibm.com/developerworks/library/l-posix1/