Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Παράλληλα Υπολογιστικά Συστήματα
Σεπτέμβριος 2019

Παραλληλοποίηση Αλγορίθμου Προσομοίωσης Μεταφοράς Θερμότητας

Γιώργος Κατσογιάννης-Μεϊμαράκης
sdi1400065@di.uoa.gr

Γιάννης Χήρας
sdi1400225@di.uoa.gr

Περιεχόμενα

1 Εισαγωγή

HEAT2D is based on a simplified two-dimensional heat equation domain decomposition. The initial temperature is computed to be high in the middle of the domain and zero at the boundaries. The boundaries are held at zero throughout the simulation. During the time-stepping, an array containing two domains is used; these domains alternate between old data and new data. At each time step, worker processes must exchange border data with neighbors, because a grid point's current temperature depends upon its previous time step value plus the values of the neighboring grid points.

Για κάθε τετράγωνο του πλέγματος η τιμή της θερμοκρασίας στο επόμενο time-step υπολογίζεται από την εξής εξίσωση:

$$\begin{aligned} u'[x][y] = u[x][y] \\ &+ c_x * (u[x+1][y] + u[x-1][y] - 2 * u[x][y]) \\ &+ c_y * (u[x][y+1] + u[x][y-1] - 2 * u[x][y]) \end{aligned}$$

Όπου $c_x = c_y = 0.1$ σταθερές.

2 Μεταγλώττιση & Εκτέλεση

2.1 Μεταγλώττιση

Τα προγράμματα μεταγλωττίζονται με τη χρήση του αρχείου Makefile και συγκεκριμένα τις εντολές:

- `make`
  - Για τη δημιουργία και όλων των εκτελέσιμων
- `make lsh`
  - Για τη δημιουργία μόνο του εκτελέσιμου lsh
- `make cube`
  - Για τη δημιουργία μόνο του εκτελέσιμου cube
- `make cluster`
  - Για τη δημιουργία μόνο του εκτελέσιμου cluster

2.2 Εκτέλεση

2.2.1 LSH

```
./lsh -d <input file> -q <query file> -k <int> -L <int> -o <output file>
```

2.2.2 Παράμετροι LSH

- L είναι το πλήθος των hash tables που θα δημιουργηθούν
- k είναι το πλήθος των συναρτήσεων κατακερματισμού h_i ανά hash table

2.3 Input File

Τα αρχεία εισόδου όλων των προγραμμάτων του project πρέπει να έχουν την ακόλουθη μορφή:

```
x0  0   16  35  5   32  31  14  10  11  78  55  10  45  83  11  6   14  57  ...
x1  14  35  19  20  3   1   13  11  16  119 85  5   0   5   24  26  0   27  ...
x2  0   1   5   3   44  40  20  14  10  100 63  7   44  47  9   6   7   70  ...
x3  12  47  14  25  2   3   4   7   14  122 90  7   0   0   6   14  0   24  ...
...
```

```
x0,0,16,35,5,32,31,14,10,11,78,55,10,45,83,11,6,14,57,...
x1,14,35,19,20,3,1,13,11,16,119,85,5,0,5,24,26,0,27,...
x2,0,1,5,3,44,40,20,14,10,100,63,7,44,47,9,6,7,70,...
x3,12,47,14,25,2,3,4,7,14,122,90,7,0,0,6,14,0,24,...
...
```

3 Εργαλεία Ανάπτυξης Project

3.1 Version Control (Git/Github)

Για την καλύτερη διαχείρηση των εκδόσεων του κώδικα και των αλλαγών χρησιμοποιείται το πρόγραμμα git και η πλατφόρμα Github.

3.2 Unit Testing (CppUnit)

Για τον έλεγχο της καλής λειτουργίας των κομματιών του project χρησιμοποιείται η βιβλιοθήκη CppUnit.

Η εγκατάσταση του CppUnit γίνεται με την εντολή:

```
sudo apt-get install libcppunit-dev
```

Η μεταγλώττιση των test γίνεται με το αρχείο Makefile και την εντολή:

```
make test
```

Στη συνέχεια τρέχοντας το εκτελεσίμο test που δημιουργείται στον φάκελο bin βλέπουμε αν υπήρχε κάποιο σφάλμα.

4 Οργάνωση Αρχείων & Φακέλων

```
project
├── bin
│   └── ...
├── data
│   └── ...
├── doc
│   ├── readme
│   │   └── ...
│   └── ...
├── include
│   └── ...
├── obj
│   └── ...
```

Ο κώδικας οργανώνεται σε διαφορετικά αρχεία ανάλογα με το σκοπό και τη λειτουργικότητά του. Συγκεκριμένα στα εξής αρχεία:

4.1 Οργάνωση Φακέλων

4.1.1 Φάκελος bin

Περιέχει τα τελικά εκτελέσιμα αρχεία που παράγονται με τη μεταγλώττιση του κώδικα.