ARISTOTLE
UNIVERSITY OF
THESSALONIKI

Κατανεμημένα και Διαδικτυακά Συστήματα
Εργασία OpenMP-OpenMPI

Αστέριος Χουλιάρας ΑΕΜ:2428
Κωνσταντίνος Μπένος ΑΕΜ:2384

5 Απριλίου 2016

https://github.com/asterisch/kds_Project

1 Περιγραφή

1.1 Γενικά

Το θέμα αυτής της εργασίας είναι η μελέτη της αποδοτικότητας συστοιχίας υπολογιστών (cluster) σε συγκεκριμένους υπολογισμούς πάνω σε αριθμητικά δεδομένα.Τα δεδομένα αυτά προέρχονται από μετρήσεις σε επιταχυντές του CERN, όπου συγκρουόμενα σωματίδια αφήνουν ένα αποτύπωμα στον τρισδιάστατο χώρο.Σύμφωνα με εκτιμήσεις αναμένονται να υπάρχουν 5.000.000 συγκρούσεις ανά δευτερόλεπτο σε μία μόνο συγκεκριμένη περιοχή του επιταχυντή.Στόχος μας είναι να αναπτύξουμε αλγορίθμους που να επεξεργάζονται αυτά τα δεδομένα με τον πιο αποδοτικό τρόπο, ώστε να εξάγουμε το πλήθος των συγκρούσεων που ανιχνεύθηκαν σε μια περιοχή και να μετρήσουμε την απόδοση αυτών στη συστοιχίας, ώστε να μπορούμε να συμπεράνουμε αν μπορεί να ανταπεξέλθει σε φόρτο 5.000.000/δευτερόλεπτο.

1.2 Generator

Αφού δεν έχουμε πραγματικά πρόσβαση στο CERN για αληθινά δεδομένα, σημαίνει ότι αρχικά δημιουργήσαμε μια προσομοίωση παραγωγής των δεδομένων (generator) και αφού δεν γνωρίζουμε με ποια πιθανότητα συγκρούονται κάποια σωματίδια,παράγουμε τυχαίες συντεταγμένες συγκρούσεων σε ένα συγκεκριμένο εύρος.Τα δεδομένα αυτά αποθηκεύονται σε ένα αρχείο στην ίδια διαδρομή με το εκτελέσιμο πρόγραμμα και αποτελούνται από τριάδες αριθμών κινητής υποδιαστολής διπλής ακρίβειας με έξι δεκαδικά ψηφία.Για λόγους φορμαλισμού κάθε γραμμή αυτού του αρχείου αποτελείται από 31 χαρακτήρες συνολικά.

1.3 Examiner

Για την επεξεργασία των δεδομένων (examiner) υλοποιήσαμε τέσσερις διαφορετικές εκδοχές του ίδιου προγράμματος,ώστε να συγκρίνουμε την απόδοση τους.Η πρώτη, αφορά την σειριακή εκτέλεση του προγράμματος από έναν υπολογιστή,όπου τα δεδομένα εξετάζονται στη σειρά το ένα μετά το άλλο και ο χρόνος εκτέλεσης εξαρτάται αποκλειστικά από τρόπο δρομολόγη-

σης της διεργασίας από το σύστημα και από τον όγκο των δεδομένων.Η δεύτερη,εισάγει την έννοια του παραλληλισμού εργασιών και αφορά την χρήση νημάτων του επεξεργαστή,όπου ο φόρτος διαμοιράζεται στα νήματα και το καθένα εκτελεί ένα κομμάτι ανεξάρτητο από τα υπόλοιπα.Η τρίτη υλοποίηση έχει να κάνει με την δημιουργία πολλαπλών διεργασιών,όπου η κάθε μια αναλαμβάνει ένα κομμάτι των δεδομένων ανεξάρτητο πάλι από τις άλλες, ενώ η τελευταία συνδυάζει τις δύο παραπάνω μεθόδους πολλαπλών διεργασιών και νημάτων.

2 Βασικά Σημεία του Κώδικα

Αρχικά ελέγχεται η είσοδος του προγράμματος που αφορά τις διάφορες παραμέτρους και αν είναι στη σωστή μορφή ρυθμίζονται οι αντίστοιχες μεταβλητές,αλλιώς εμφανίζεται μήνυμα υποδεικνύοντας τη σωστή χρήση.Στη συνέχεια,η master διεργασία μετρά τις σειρές του αρχείου δεδομένων ώστε να κατανέμει το συνολικό φόρτο στις υπόλοιπες διεργασίες.Κάθε διεργασία αποκτά ένα δείκτη μέσα στο αρχείο,που υποδηλώνει τη θέση από την οποία θα ξεκινήσει την επεξεργασία.Από κάθε νήμα και κάθε διεργασία χρησιμοποιείται μόνο ένας πίνακας τριών θέσεων, όπου κάθε φορά διαβάζονται ανά τριάδες οι συντεταγμένες από το αρχείο και ελέγχονται μέσω αυτού.Αυτό συνεισφέρει στην εξοικονόμηση μνήμης αφού ο φόρτος της κύριας μνήμης για την επεξεργασία συνολικά είναι:

(πλήθος νημάτων)(πλήθος διεργασιών)*(μέγεθος float).*

Κάθε φορά που ένα νήμα μιας διεργασίας εντοπίζει μια τριάδα μέσα στο εύρος ενδιαφέροντος, αυξάνει μια μεταβλητή μετρητή και στο τέλος αυτές συναθροίζονται για να δώσουν το συνολικό πλήθος συντεταγμένων.Τέλος, ο χρόνος εκτέλεσης αφορά την βασική λειτουργία του προγράμματος που είναι η επεξεργασία των δεδομένων και βάση αυτού του χρόνου προφανώς υπολογίζεται και ο ρυθμός επεξεργασίας.

Σε γενικές γραμμές φροντίσαμε ο κώδικας να είναι επαρκώς σχολιασμένος, ώστε κάθε κομμάτι του να είναι όσο το δυνατόν πιο κατανοητό και τηρήσαμε κάποιες βασικές αρχές συγγραφής κώδικα, ώστε να είναι και εύκολος στην ανάγνωση.

2.1 Βασικά σημεία με τη χρήση OpenMP

1. *Διαχείριση της παραμέτρου του προγράμματος ώστε να παραχθούν τα αντίστοιχα νήματα από την διεργασία*

2. *Διαχείριση κοινών και ιδιωτικών μεταβλητών.Κάποιες πρέπει να μοιράζονται τα νήματα για εξοικονόμηση μνήμης και ταχύτητα,όπως η μεταβλητή μέτρησης των συντεταγμένων στην περιοχή ενδιαφέροντος και κάποιες πρέπει να είναι ιδιωτικές,όπως ο φόρτος και ο πίνακας που αποθηκεύονται οι συντεταγμένες.*

3. *Προσοχή στην ταυτόχρονη τροποποίηση μιας κοινόχρηστης μεταβλητής,όπως ο μετρητής,ώστε κάθε φορά ένα και μόνο νήμα να αλλάζει την τιμή της.*

4. *Επιλογή δυναμικής κατανομής φόρτου στα νήματα με στόχο την αποδοτικότητα.*

2.2 Βασικά σημεία με τη χρήση OpenMPI

1. *Τμηματοποίηση του κώδικα,ώστε οι διεργασίες να μην έχουν αχρησιμοποίητες μεταβλητές και να εκτελούν άσκοπες πράξεις.Για παράδειγμα αποτελεί η επεξεργασία των παραμέτρων του προγράμματος όπου αρκεί να γίνει μόνο σε μια διεργασία και να σταλθεί στις υπόλοιπες καθώς και η μέτρηση του χρόνου εκτέλεσης,αφού η master διεργασία περιμένει να ολοκληρωθούν όλες οι υπόλοιπες προτού τερματίσει.*

2. *Διαμερισμός του φόρτου ανάλογα με το πλήθος των διεργασιών.Χρήση ενός δείκτη στο αρχείο,όπου κατά την έναρξη της διεργασίας απέχει από την αρχή του αρχείου τόσα byte όσα η τάξη της επί τον φόρτο διεργασιών μικρότερης τάξης.Έτσι κάθε διεργασία επεξεργάζεται ένα σύνολο δεδομένων που τελειώνει εκεί που αρχίζει το σύνολο κάποιας άλλης διεργασίας.*

3. *Διαχείριση της παραμέτρου του προγράμματος ώστε να χρησιμοποιηθούν ο αντίστοιχος αριθμός από διεργασίες.Στην περίπτωση όπου ζητηθούν διεργασίες περισσότερες άπ' οσες δηλώθηκαν στον εκτελεστή προγράμματος mpi (π.χ. mpirun,mpiexec,..),η είσοδος αυτή αγνοείται.*

2.3 CUDA

Για τις παραμέτρους που δίνει ο χρήστης υπάρχουν default τιμές στον κώδικα, πχ block size 512 και threads όσα μπορεί να υποστηρίξει συνολικά η κάρτα γραφικών, και αν υπάρχει λάθος σε κάποιο argument το πρόγραμμα τερματίζει με κωδικό λάθους. Ξεκινώντας αρχικοποιούμε τις μεταβλητές για το host και το device και δεσμεύουμε κατάλληλα τον χώρο στην μνήμη. Στη συνέχεια αρχικοποιούμε την τιμή του μετρητή συντεταγμένων στην περιοχή αναφοράς στην κάρτα γραφικών και μπαίνουμε στην κύρια επανάληψη του προγράμματος. Τα δεδομένα διαβάζονται από το αρχείο σύμφωνα με την τιμή του argument 'threads' και φορτώνονται στην μνήμη cpu. Αμέσως μετά τα δεδομένα αντιγράφονται στην global gpu memory και καλείται ο kernel ο οποίος επεξεργάζεται τα δεδομένα πάνω στην gpu. Παρατηρούμε ότι όσο μεγαλύτερος είναι ο αριθμός των δεδομένων που φορτώνονται με τη μία στη gpu μνήμη τόσο μεγαλύτερη είναι η αποδοτικότητά της στην ταχύτητα υπολογισμών.Αυτό συμβαίνει γιατί όταν αντιγράφουμε σταδιακά τα δεδομένα από τη μνήμη cpu στη μνήμη gpu έχουμε καθυστέρηση στη μεταφορά.Στην τελευταία επανάληψη φορτώνονται τα εναπομείναντα δεδομένα από τον διαχωρισμό που είχε κάνει αρχικά η cpu σύμφωνα με την παράμετρο 'threads' και το μέγεθος του αρχείου. Τέλος όταν ολοκληρωθεί η επεξεργασία στην κάρτα γραφικών αντιγράφουμε την τιμή του μετρητή συντεταγμένων από την global memory της gpu στην cpu memory, εμφανίζουμε τα αποτελέσματα και αποδεσμεύουμε μνήμη μεταβλητών στην GPU και στη CPU .

2.3.1 Παρατηρήσεις

1. *Οι συντεταγμένες που διαβάζονται αποθηκεύονται σε μονοδιάστατο πίνακα στην μνήμη της CPU.*

2. *Όταν μεταφερθούν στον μονοδιάστατο της GPU κάθε Thread είναι υπεύθυνο για τη δική του τριάδα (=1 συντεταγμένη [x,y,z]).Οπότε υπολογίζουμε για κάθε νήμα την θέση του πρώτου αριθμού της τριάδας στον πίνακα και μέσω ενός offset εξασφαλίζουμε ότι το κάθε νήμα εξετάζει μια και μοναδική συντεταγμένη.*

3. *Μέσω της λειτουργίας atomicAdd εξασφαλίζεται η πρόσβαση και η τροποποίηση του μετρητή από ένα και μοναδικό νήμα, ώστε να μην υπάρξουν ταυτόχρονες τροποποιήσεις.*

4. *Όσο πιο πολλά threads καλούμε μέσω της παραμέτρου τόσο πιο γρήγορα αυτά επεξεργάζονται αντίστοιχα δεδομένα, γιατί:*
   1) Περισσότερα δεδομένα μεταφέρονται με μια επανάληψη από CPU σε GPU μνήμη,που σημαίνει κέρδος σε χρόνο.
   2) Η GPU διαχειρίζεται τα blocks από threads που έχει μέσα στο grid πιο αποδοτικά.

5. *Δίνεται η δυνατότητα στον χρήστη να καλεί το πρόγραμμα ορίζοντας το μέγεθος του block νημάτων στην GPU και το πλήθος τους.*

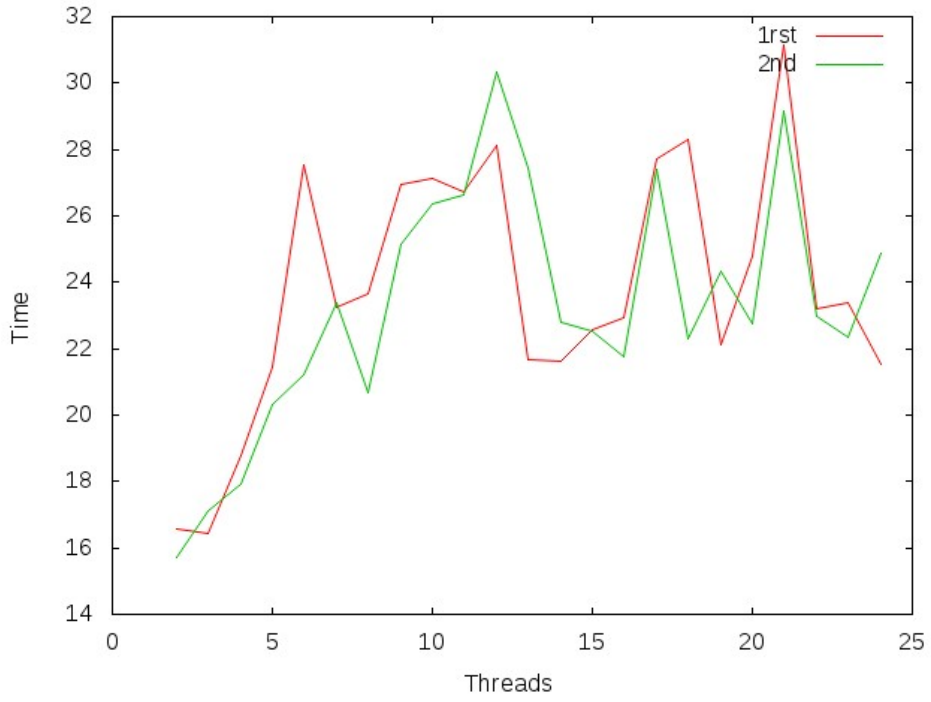
```
nvcc src\examine.cu -o examine.exe
examine.cu
   Creating library examine.lib and object examine.exp



examine.exe -1 -1 datafile -1 -1
[+] GPU-model: GeForce GT 740M  Total GPU memory 2048 MB
[!] You are trying to allocate 0 MBs of memmory on CPU-RAM and GPU-GlobalMem
[+] Launching 4096 GPU-Threads with BlockSize 512
[+] Working...
```

Σχήμα 1: Compile & Run commands

3 Χρόνοι και Αποτελέσματα

Ο Δοκιμές των προγραμμάτων έγιναν στη συστοιχία του Πανεπιστημίου Δυτικής Μακεδονίας.Τα δεδομένα ήταν σ' ένα κοινό αρχείο, ώστε να είναι δίκαιες οι μετρήσεις,αποτελούμενο από 15.000.000 τριάδες συντεταγμένων. Τα γραφήματα έγιναν μέσω του gnuplot, σε δεδομένα που προήλθαν από δοκιμές στο cluster με τη ρύθμιση Nodes=1:ppn=24.

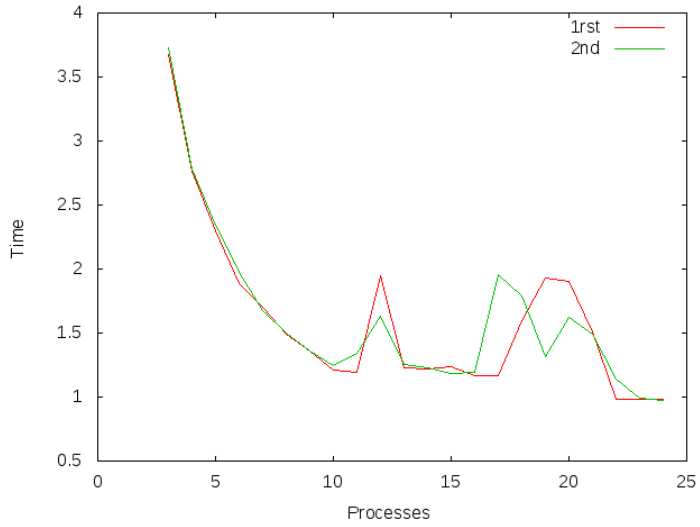
3.1 OpenMP



Σχήμα 2: Χρόνος εκτέλεσης ανάλογα με τα νήματα

Παρατηρούμε ότι τα πολλά νήματα σε μία μόνο διεργασία εισάγουν κάποια καθυστέρηση.Αυτό εν μέρη οφείλεται και στη δυσκολία ταυτόχρονης πρόσβασης στο αρχείο από πολλά νήματα.

3.2 OpenMPI



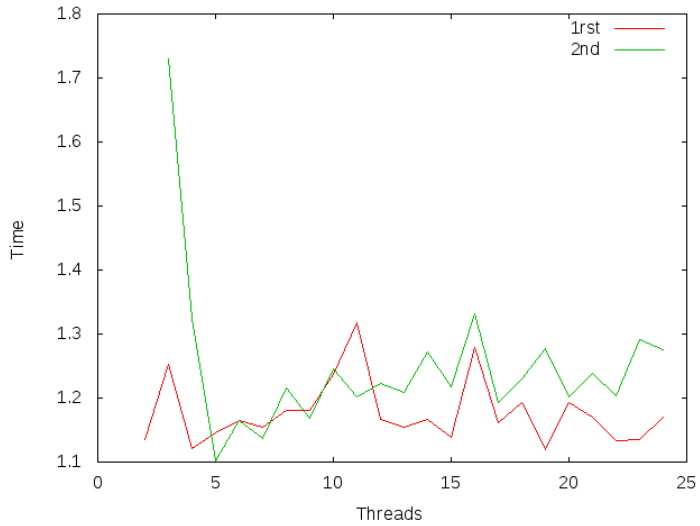
Σχήμα 3: Χρόνος εκτέλεσης ανάλογα με τις διεργασίες

Με την χρήση του πρωτοκόλλου OMPI πετυχαίνουμε πολύ καλές επιδώσεις,αφού ο διαχωρισμός δεδομένων-πράξεων γίνεται αποτελεσματικά σ' όλες τις διεργασίες.Με λίγες διεργασίες η απόδοση μειώνεται δραματικά, ενώ όσο αυξάνονται αυξάνεται και η απόδοση.

```
asteriosc@ubuntu:~/kds
Time         Processes
0.573107601    24
0.605796263    24
0.632824302    24
1.57750325     12
1.80745639     12
1.92423214     12
1.929254338     6
2.249100359     6
2.330906201     6
```

Σχήμα 4: Nodes=4 : ppn=6

3.3 OpenMP & OpenMPI



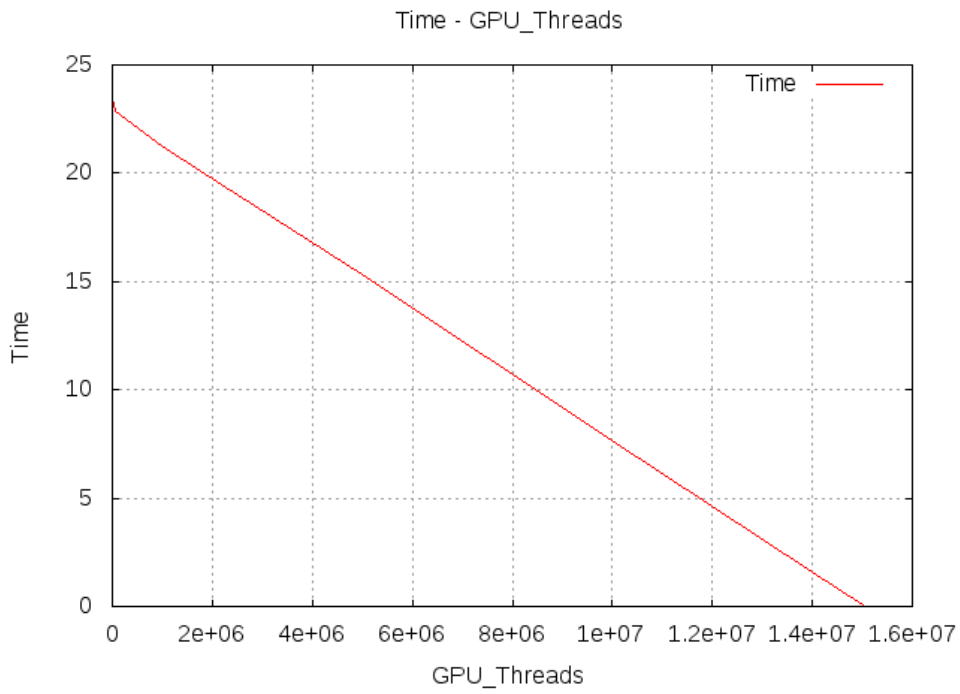
Σχήμα 5: Σταθερές 24 διεργασίες και μεταβλητά νήματα

Ο συνδυασμός και των δύο έχει καλά αποτελέσματα,ανάλογα με τις διεργασίες και τα νήματα.

```
asteriosc@ubuntu:~/kds_Project/Te
Time          Processes   Threads
1.105249002   24          9
1.117976406   24          15
1.151299264   24          2
1.185384120   24          5
1.211295250   24          13
1.213725879   24          21
1.230478549   24          6
1.242032593   24          16
1.314379126   24          7
1.317386520   24          18
1.325391030   24          14
1.389583935   24          17
1.452657233   24          23
1.460765256   24          3
1.480314358   24          4
1.530748245   24          22
1.539000364   24          20
1.543205317   24          19
1.567834688   24          10
1.573400215   24          8
```

Σχήμα 6: Nodes=4 : ppn=6

3.4 CUDA



Σχήμα 7: Χρόνος εκτέλεσης kernel σε αναλογία με το πλήθος των GPU Threads

Ο αριθμός των GPU-Threads που ζητούνται από τη κάρτα γραφικών παίζει καθοριστικό ρόλο στην απόδοση του προγράμματος, άρα και στην ταχύτητα ανάλυσης των δεδομένων.Παρατηρούμε ότι ο χρόνος εκτέλεσης μειώνεται γραμμικά σε σχέση με το πλήθος των νημάτων,όμως όταν αυτά ξεπεράσουν σε πλήθος τα δεδομένα δεν υπάρχει περαιτέρω βελτίωση στην απόδοση.Τελικά, συμπεραίνουμε ότι η χρήση των επεξεργαστών της κάρτας γραφικών προσφέρει άριστη παραλληλοποίηση σε υπολογισμούς,αφού εστιάζει στην διεκπεραίωση όσο το δυνατόν περισσότερων εργασιών ταυτόχρονα, σε αντίθεση με την CPU, όπου η διεκπεραίωση μιας εργασίας γρήγορα είναι πιο σημαντική από το συνολικό πλήθος τους.