Ετήσια Τεχνική Έκθεση

Έτος 2013

ΘΑΛΗΣ – Πολυτεχνείο Κρήτης

Πλατφόρμα προηγμένων μαθηματικών μεθόδων και λογισμικού για την επίλυση προβλημάτων πολλαπλών πεδίων (multi physics, multidomain) σε σύγχρονες υπολογιστικές αρχιτεκτονικές: Εφαρμογή σε προβλήματα Περιβαλλοντικής Μηχανικής και Ιατρικής (MATENVMED- MIS 379416)

Δράση 3.2

ΥΛΟΠΟΙΗΣΗ ΣΕ FPGAs ΚΑΙ ΠΟΛΥΠΥΡΗΝΑ ΣΥΣΤΗΜΑΤΑ (MULTICORES / MANYCORES)

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Πίνακας Περιεχομένων

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

1. Σκοπός

Η έκθεση για το 2013 παρουσιάζει την πρόοδο της Δράσης 3.2 στις δύο πλατφόρμες CPU/GPU και FPGA δίδοντας έμφαση στην ανάπτυξη εργαλείων που θα χρησιμοποιηθούν για υλοποίηση των elliptic PDE solvers σε multicore CPU και GPU.

Σε αυτήν την έκθεση παρουσιάζουμε αρχικά τα αποτελέσματα της εκτέλεσης ενός προγράμματος επίλυσης Μερικών Διαφορικών Εξισώσεων (Partial Differential Equations, PDEs) με την χρήση τεχνικών Monte Carlo. Η υλοποίηση του προγράμματος επίλυσης των PDEs γίνεται σε μια πολυπύρηνη CPU και σε 4 GPUs χρησιμοποιώντας OpenCL [1].

Επίσης παρουσιάζουμε ένα καινοτόμο πλαίσιο το οποίο αυτοματοποιεί την διαχείριση δεδομένων (data management) από ένα σύστημα χρόνου εκτέλεσης (runtime system, RTS) σε ένα ετερογενές επεξεργαστικό σύστημα. Το πλαίσιο αυτό χρησιμοποιεί compile-time ανάλυση βασισμένη στο πολυεδρικό μοντέλο (polyhedral model) με σκοπό να συνδέσει υπολογισμούς με τα δεδομένα που αυτοί οι υπολογισμοί παράγουν και καταναλώνουν. Τα αποτελέσματα της πολυεδρικής ανάλυσης (polyhedral analysis) χρησιμοποιούνται στη συνέχεια από το RTS για αυτοματοποίηση της διαχείρισης δεδομένων. Πέρα από το ότι απαλλάσσει τον προγραμματιστή από το βάρος της διαχείρισης δεδομένων, το πλαίσιο μας δίνει τη δυνατότητα αυτόματου διαχωρισμού των δεδομένων στις μνήμες διαφορετικών επεξεργαστών ανάλογα με την υπολογιστική τους ισχύ και το μέγεθος της μνήμης που περιέχουν.

Τα πειραματικά αποτελέσματα μας δείχνουν ότι το πλαίσιο αυτό διευκολύνει την χρήση όλων των ετερογενών πόρων (resources) του συστήματος με πολύ μικρή επιβάρυνση της τάξης του 1.24% σε σχέση με το να γίνει η κατανομή των δεδομένων από τον προγραμματιστή.

2. Υλοποίηση PDEs σε ετερογενές σύστημα CPU/GPU με την χρήση πολυεδρικού μοντέλου

2.1. Εισαγωγή

Τα ετερογενή συστήματα γίνονται ολοένα και πιο δημοφιλή στην υπολογιστική υψηλών επιδόσεων κυρίως λόγω της δυνατότητας τους να συνδυάζουν υψηλή ταχύτητα με σχετικά χαμηλή κατανάλωση ισχύος. Το σημαντικότερο πρόβλημα τους είναι ότι επιβαρύνουν τον προγραμματιστή όχι μόνο με την εύρεση και εκμετάλλευση του παραλληλισμού σε μια εφαρμογή αλλά επίσης και με την ευθύνη της μεταφοράς δεδομένων από τον χώρο διευθύνσεων ενός επεξεργαστή (πχ CPU) στον χώρο διευθύνσεων ενός διαφορετικού επεξεργαστή (πχ GPU). Ακόμα χειρότερα, η μεγάλη διαφοροποίηση μεταξύ αρχιτεκτονικών και μνημών στα ετερογενή συστήματα

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

αναγκάζουν πολλές φορές τον προγραμματιστή να δημιουργήσει πολλές διαφορετικές εκδοχές ενός κώδικα για να μπορέσει να καλύψει όλες τις περιπτώσεις.

Για να υλοποιήσουμε τον αλγόριθμο επίλυσης PDEs μέσω τεχνικών Monte Carlo, αποφασίσαμε να χρησιμοποιήσουμε το προγραμματιστικό μοντέλο OpenCL που στοχεύει ακριβώς τα ετερογενή συστήματα. Επιπλέον δημιουργούμε ένα πλαίσιο που συνδυάζει compile-time στατική ανάλυση με run-time δυναμική ανάλυση της ροής των δεδομένων μιας εφαρμογής για να αυτοματοποιήσει την διαχείριση δεδομένων σε ετερογενή συστήματα CPU και GPU. Πέρα από την αυτόματη διαχείριση δεδομένων, το πλαίσιο αυτό βοηθάει στην κατανομή των υπολογισμών στους υπάρχοντες επεξεργαστές λαμβάνοντας υπόψιν του την υπολογιστική ισχύ και το ενεργειακό αποτύπωμα κάθε επεξεργαστή, βελτιστοποιώντας κατά αυτόν τον τρόπο την χρήση της υπολογιστικής ισχύς και των μνημών του ετερογενούς συστήματος.

Η μεθοδολογία μας εισάγει μία αμελητέα επιβάρυνση στον χρόνο εκτέλεσης κατά 1.24% σε σχέση με την περίπτωση να γίνει η διαχείριση δεδομένων μόνο από τον επεξεργαστή. Στην συνέχεια της Παραγράφου 2, θα εισάγουμε βασικές έννοιες του πολυεδρικού μοντέλου στην Παρ. 2.2, θα περιγράψουμε την μεθοδολογία του πλαισίου μας στην Παρ. 2.3, και θα παρουσιάσουμε τα τελικά αποτελέσματα της μεθόδου μας στην Παρ. 0.

2.2. Πολυεδρικό Μοντέλο

Το πολυεδρικό μοντέλο είναι μαθηματικό πλαίσιο που χρησιμοποιείται κυρίως για ανάλυση βρόγχων (loops) για βελτιστοποίηση της μεταγλώττισης (compilation) σε ένα πρόγραμμα. Η μέθοδος του πολύεδρου θεωρεί ότι κάθε επανάληψη του loop (loop iteration) μπορεί να αναπαρασταθεί από ένα σημείο μέσα σε μαθηματικά αντικείμενα που ονομάζονται πολύεδρα (ή πολύτοπα). Μετασχηματισμοί που εφαρμόζονται πάνω σε πολύεδρα μεταφράζονται σε μετασχηματισμούς του πηγαίου κώδικα του προγράμματος μας. Η πολυεδρική ανάλυση αποτελεί ένα πολύτιμο εργαλείο για βελτιστοποίηση κώδικα μέσω του compiler λόγω της ευχέρειας να αλλάζει την μορφή των loops.

Ένα n-πολύεδρο είναι ένα γεωμετρικό αντικείμενο με επίπεδες πλευρές στον N-διάστατο χώρο. Το πεδίο ορισμού D του πολυέδρου είναι η τομή ενός πεπερασμένου συνόλου από κλειστά ημίσεια διαστήματα (half spaces). Αυτή η αναπαράσταση μπορεί να γραφεί και ως ένα σύστημα από εξισώσεις και ανισώσεις:

$$D: \{x \in Q^n | Ax = b, Cx \geq D\}$$

Η ανάλυση με την μέθοδο του πολυέδρου αναπαριστά τα loops καθώς και τις εντολές μέσα στα loops ως πολύεδρα. Χρησιμοποιείται ευρέως σε δημοφιλής compilers όπως ο LLVM και ο gcc. Μέσω της δημιουργίας πολυέδρων, ένας compiler μπορεί να εντοπίσει εξαρτήσεις μεταξύ εντολών (data dependencies), να δημιουργήσει βελτιστοποιημένους χρονοπρογραμματισμούς εντολών (instruction schedules) και να ανακαλύψει παραλληλισμό μεταξύ εντολών ή επαναλήψεων ενός loop.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Παράδειγμα Ανάλυσης Πολυέδρου. Θα δείξουμε ένα απλό παράδειγμα ανάλυσης χρησιμοποιώντας την μέθοδο του πολυέδρου και αναφερόμενοι στον κώδικα της Εικόνα 1.

```
for (i=2; i <= min(M, -1+N+2), i++) {
   S1(i);
}
```

Εικόνα 1. Απλός κώδικας για το παράδειγμά μας.

Η συνθήκη που ελέγχει τον τερματισμό του loop αντιστοιχεί στον περιορισμό:

$2 \leq i \leq \min(M, -1 + N + 2)$ ο οποίος μπορεί να γραφεί και ως:

$2 \leq i \leq M$, και $2 \leq i \leq N + 1$.

Ένα πολύεδρο μπορεί να αναπαρασταθεί από έναν πίνακα με *(1+Output+Input+Parameters+1)* στήλες και τόσες γραμμές όσος και ο αριθμός των περιορισμών που δημιουργούν το πολύεδρο. Η πρώτη στήλη δείχνει εάν η αντίστοιχη γραμμή περιγράφει ισότητα ή ανισότητα (τιμή 0 ή 1 , αντίστοιχα). Τα *Outputs* αντιστοιχούν σε δείκτες πινάκων, και στο συγκεκριμένο παράδειγμα δείχνουν ότι το εύρος τιμών της μεταβλητής *i*, που είναι το μοναδικό output είναι η ίδια η τιμή του *i*. Η τρίτη στήλη αντιστοιχεί στον μοναδικό iterator *i* του loop, ενώ οι επόμενες δύο στήλες αντιστοιχούν στις παραμέτρους. Η τελευταία στήλη αποθηκεύει το σταθερό κομμάτι των περιορισμών. Ο πίνακας που αντιπροσωπεύει το πολύεδρο του κώδικά μας είναι ο παρακάτω:

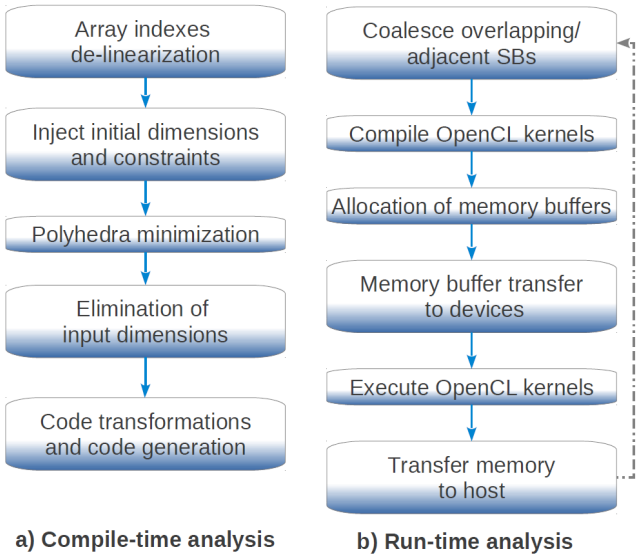
$$\begin{pmatrix} 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & -2 \\ 1 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 1 \end{pmatrix}$$

2.3. Πλαίσιο ανάλυσης διαχείρισης δεδομένων

Σε αυτήν την παράγραφο θα εξηγήσουμε την μεθοδολογία εύρεσης του μεγέθους της μνήμης που απαιτεί ένας συγκεκριμένος υπολογισμός και τον τρόπο με τον οποίο μπορούμε να αυτοματοποιήσουμε μεταφορές δεδομένων σε μία ετερογενή πλατφόρμα. Αυτή η τεχνική μπορεί επίσης να χρησιμοποιηθεί για την παραλληλοποίηση ενός αλγορίθμου πιθανώς σε τμήματα διαφορετικού βαθμού διακριτότητας (granularity) από ότι αρχικά έχει καθορίσει ο προγραμματιστής.

Η πρώτη φάση της πολυεδρικής ανάλυσης λαμβάνει χώρα κατά την διάρκεια του compilation. Στοχεύει στο να δημιουργήσει μια σειρά από παραμετρικές εξισώσεις που να υπολογίζουν το διάστημα των θέσεων μνήμης που μπορούν να προσπελασθούν από κάθε κομμάτι του κώδικα. Κατά την διάρκεια εκτέλεσης του κώδικα, η δεύτερη φάση διαχωρίζει αυτόματα τα δεδομένα και τα μεταφέρει με έναν βέλτιστο τρόπο μεταξύ των επεξεργαστικών στοιχείων. Η Εικόνα 2 δείχνει την ροή και των δύο φάσεων του αλγορίθμου μας.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 2. Η ροή του υβριδικού αλγορίθμου μας

Στατική φάση (compilation).

Η ανάπτυξη της στατικής φάσης έγινε με την βοήθεια εργαλείων όπως το Polyhedral Extraction Tool (PET) [3], το ISL [2] και το PolyLib [4]. Το PET παίρνει σαν είσοδο τον πηγαίο κώδικα και δημιουργεί το πολυεδρικό μοντέλο για τα loops του κώδικα αυτού. Η ISL είναι μία βιβλιοθήκη εργαλείων για καλύτερη αναπαράσταση πολυέδρων, ενώ η PolyLib είναι μία βιβλιοθήκη που παρέχει εργαλεία για την επεξεργασία πολυέδρων.

Το αποτέλεσμα της πρώτης φάσης του αλγόριθμού μας είναι ένα σύνολο παραμετροποιημένων διαστημάτων για κάθε πίνακα (array) που προσπελαύνεται μέσα στο loop. Κάθε τέτοιο διάστημα περιέχει επίσης πληροφορίες για τον τύπο της προσπέλασης (read/write). Τα στάδια της φάσης αυτής που φαίνονται και στην Εικόνα 2 είναι τα εξής:

1. Array index de-linearization. Η ανάλυση μας αφορά κάθε εντολή προγράμματος μέσα σε loops που προσπελαύνει μονοδιάστατους και δισδιάστατους πίνακες εφόσον οι προσπελάσεις είναι του τύπου *array[row*width+column]*. Λόγω της παρουσίας της παραμέτρου *width,* που αναπαριστά τον αριθμό των στηλών του πίνακα, το εργαλείο PET δεν μπορεί να ολοκληρώσει την ανάλυση επειδή θεωρεί ότι η αντίστοιχη προσπέλαση δεν είναι συσχετισμένη (affine). Αντιμετωπίζουμε αυτό το πρόβλημα με το να επεκτείνουμε το εργαλείο PET θεωρώντας τους δείκτες *row* και *column* σαν ανεξάρτητους δείκτες και το *width* σαν extra παράμετρο.
2. Inject initial dimensions and constraints. Ο κώδικας ενός παράλληλου πυρήνα (kernels) σε OpenCL ουσιαστικά περικλείεται από 6 loops, τα οποία αντιστοιχούν στην εκτέλεση ενός work-item της OpenCL μέσα σε ένα work-group (τα τρία εσωτερικά loops) και στην εκτέλεση ενός work-group μέσα σε ένα πλέγμα (grid) (τα τρία εξωτερικά loops). Σε αυτό το βήμα, εισάγουμε αυτά τα extra loops στο πολυεδρικό μας μοντέλο.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

```
kernel void DoRandomWalks2D(
global float *D, global float *x, global float *result,
unsigned int num_walks, float btol, unsigned int nodes)
```

3. Polyhedral minimization. Ελαχιστοποίηση του μεγέθους του πολυέδρου μέσω της απαλοιφής περιορισμών που είναι πλεονάζοντες.
4. Elimination of input dimensions. Το επόμενο βήμα της στατικής φάσης είναι η μείωση της διάστασης των εισόδων (inputs) του πολυέδρου. Πολλές είσοδοι έχουν σταθερή τιμή κατά την διάρκεια της εκτέλεσης του κώδικα (πχ η παράμετρος *width*) και οι προσπελάσεις της μνήμης λόγω αυτών των εισόδων μπορούν να υπολογισθούν κατά την διάρκεια του compilation. Μετά το τέλος της φάσης αυτής, όλα τα πολύεδρα αποτελούνται από στήλες που αντιστοιχούν σε outputs και παραμέτρους.
5. Code transformations and generation. Ο τελικός στόχος αυτής της φάσης είναι να χωρίσουμε τον αρχικό υπολογισμό σε μικρότερα κομμάτια και να καθορίσουμε για κάθε κομμάτι επακριβώς τα δεδομένα εισόδου και εξόδου που απαιτούνται για να εκτελεσθεί το κομμάτι αυτό. Αυτό απαιτεί το να ξαναγράψει το εργαλείο σε αυτήν την φάση τον αρχικό πηγαίο μας κώδικα.

Ο κώδικας της Εικόνας 3 και της Εικόνας 4 δείχνουν τον κώδικα Monte Carlo για την επίλυση μερικών διαφορικών εξισώσεων (PDEs) πριν και μετά την εφαρμογή του αλγορίθμου μας, αντίστοιχα. Με κόκκινο τονίζουμε τις αλλαγές όπου υπάρχουν.

Βασισμένος στην ανάλυση πολυέδρου, ο compiler δημιουργεί επίσης τον κώδικα για μία νέα συνάρτηση η οποία υπολογίζει τα διαστήματα της μνήμης που προσπελαύνονται από κάθε προσπέλαση μέσα στα κομμάτια του κώδικα που παράγονται σε αυτήν την φάση. Αυτή η συνάρτηση καλείται από το σ΄συστημα χρόνου εκτέλεσης (RTS) όταν κληθεί ο αντίστοιχος παράλληλος πυρήνας (kernel) προς εκτέλεση.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

```
{
 private long me = get_local_id(0), us = get_local_size(0);
 private long __group_id_x = get_group_id(0);
 /* Some variable declarations and statements omitted */
 if ( __group_id_x < nodes ) {
  _x[0] = x[__group_id_x*2];
  _x[1] = x[__group_id_x*2+1];
  for ( i=0; i<num_walks; ++i ) {
    _x[0] = x[__group_id_x*2];
    _x[1] = x[__group_id_x*2+1];
    perform_random_walks(num_walks, _x, _D);
  }
  barrier(CLK_LOCAL_MEM_FENCE);
  if ( me == 0 ) {
    d = compose_d();
    result[__group_id_x] = d;
  }
}
```

Εικόνα 3. Ο αρχικός κώδικας OpenCL για την μέθοδο Monte Carlo (MC).

```
/* ocl_offsets: Holds the offset values (Dynamic mode) */
 kernel void DoRandomWalks2D( constant const int *ocl_offsets,
 global float *D, global float *x, global float *result,
 unsigned int num_walks, float btol, unsigned int nodes)
 {
 private long me = get_local_id(0), us = get_local_size(0);
 private long __group_id_x = get_global_id(0)/us;
 /* Some variable declarations and statements omitted */
 if ( __group_id_x < nodes ) {
   _x[0] = x[(__group_id_x-x0y)*x0w -x0o];
   _x[1] = x[(__group_id_x-x1y)*x1w +1 -x1o];
   for ( i=0; i<num_walks; ++i ) {
     _x[0] = x[(__group_id_x-x2y)*x2w -x2o];
     _x[1] = x[(__group_id_x-x3y)*x3w +1 -x3o];
     perform_random_walks(num_walks, _x, _D);
   }
    barrier(CLK_LOCAL_MEM_FENCE);
   if ( me == 0 ) {
    d = compose_d();
     result[__group_id_x -result2o] = d;
   }
 }
}
```

Εικόνα 4. Ο κώδικας OpenCL για την μέθοδο Monte Carlo μετά την εφαρμογή του αλγορίθμου μας

Φάση χρόνου εκτέλεσης.

Αυτή η φάση λαμβάνει χώρα κατά την εκτέλεση του κώδικα λίγο πριν το

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

compilation του πυρήνα OpenCL που πρόκειται να εκτελεσθεί (σημ. στην OpenCL, οι πυρήνες είναι δυνατόν να γίνονται compile κατά την διάρκεια εκτέλεσης). Η συνάρτηση που παράγεται στο τέλος της προηγούμενης φάσης εκτελείται σε αυτήν την φάση με σκοπό να δημιουργήσει τις προσπελάσεις για κάθε συσκευή OpenCL. Τα στάδια της φάσης αυτής που φαίνονται και στην Εικόνα 2β είναι τα εξής:

1. Coalescing of overlapping/adjacent SBs. Επιμέρους περιοχές της μνήμης που προσπελαύνονται από κάθε κομμάτι του πυρήνα OpenCL συνενώνονται για να δημιουργήσουν μία μεγαλύτερη περιοχή με σκοπό να μειωθεί ο αριθμός των μεταφορών δεδομένων μεταξύ Host και των υπόλοιπων συσκευών (CPU ή GPU).
2. Allocation of memory buffers. Μετά την φάση της συνένωσης των επιμέρους τμημάτων μνήμης, γίνεται δυναμική δέσμευση ανάλογης ποσότητας μνήμης σε κάθε επεξεργαστικό στοιχείο (CPU ή GPU) που είναι στο σύστημα μας.
3. Memory buffer transfer to devices. Η μεταφορά δεδομένων μεταξύ του Host και των υπόλοιπων συσκευών γίνεται μόνο την πρώτη φορά ή όταν καινούργια δεδομένα έχουν δημιουργηθεί στις θέσεις μνήμης και θα πρέπει να μεταφερθούν και αυτά.

2.4. Αποτελέσματα της μεθόδου για PDEs

Ο αλγόριθμος για υλοποίηση των elliptic PDE solvers είναι ένας εναλλακτικός τρόπος επίλυσης προβλημάτων πολλαπλών πεδίων (multi-domain, multiphysics). Ο βασικός παράλληλος πυρήνας (kernel) του αλγορίθμου αυτού (Εικόνα 3) πραγματοποιεί τυχαίους περιπάτους (random walks) από το σύνορο (boundary) ενός υπόχωρου (sub-domain) στο σύνορο του μεγαλύτερου χώρου (domain). Ο σκοπός του περιπάτου είναι ο υπολογισμός των αρχικών συνθηκών του υπόχωρου για την επίλυση των διαφορικών εξισώσεων.

Μετά από ανάλυση και profiling της εφαρμογής μέσω της σουίτας Intel(TM) Vtune αναγνωρίσαμε δύο κρίσιμα σημεία της εφαρμογής, που καταναλώνουν σχεδόν πλήρως το χρόνο εκτέλεσης της εφαρμογής. Η πρώτη μέθοδος ονομάζεται ***monte_carlo*** και είναι η συνάρτηση υπεύθυνη για την παραγωγή τυχαίων μονοπατιών. Το αμέσως επόμενο πιο χρονοβόρο τμήμα της εφαρμογής είναι η επίλυση συστημάτων μέσω της μεθόδου ***conjugate gradient*** η οποία είναι υλοποιημένη στην βιβλιοθήκη *deal.II*. Αρχικά επιλέξαμε να επικεντρωθούμε στην συνάρτηση ***monte_carlo*** μιας και αυτή αποτελεί ουσιαστικά την ουσία της εφαρμογής.

Πριν παρουσιάσουμε τον λόγο επιτάχυνσης μέσω της χρήσης OpenCL Kernels βρίσκουμε σκόπιμο να αναφέρουμε τα χαρακτηριστικά που έχει η πλατφόρμα πάνω στην οποία εκτελέσαμε τις διαφορετικές εκδόσεις της εφαρμογής.

Για την πειραματική μελέτη απόδοσης χρησιμοποιήσαμε τα εξής δύο υπολογιστικά κυκλώματα:

- Επεξεργαστής γενικού σκοπού: *Intel(R) Core(TM) i7 CPU 870 (2.93GHz)*

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

- Κάρτα γραφικών: *GeForce GTX 480 (1401MHz)*

Για τα πειράματα που εκτελούνται στον Intel επεξεργαστή (αρχική έκδοση του κώδικα) χρησιμοποιούμε 8 νήματα ενώ για την εκτέλεση στην κάρτα γραφικών χρησιμοποιούμε N*768 νήματα όπου N ο αριθμός των σημείων στα οποία θα εφαρμοστεί η μέθοδος ***monte_carlo***. Επειδή η συνολική επιτάχυνση του προγράμματος διαμορφώνεται σε σχέση με το μέγεθος του προβλήματος που τίθεται για επίλυση παρουσιάζουμε μερικές ενδεικτικές εκτελέσεις στον παρακάτω πίνακα:

Χρόνος εκτέλεσης CPU (Διάσταση Χώρου)	**Χρόνος εκτέλεσης GPU / Χρονοβελτίωση**	**Μέσο σφάλμα CPU**	**Μέσο Σφάλμα GPU**
100 secs ***(2D)***	*34 secs /* ***2.94x***	*0.0002041409246*	*8.811696406e-05*
241 secs ***(3D)***	*35 secs /* ***6.89x***	*0.01288890583*	*0.01290900319*
761 secs ***(2D)***	*85 secs /* ***8.95x***	*0.0002024040681*	*2.788477219e-05*
2026 secs ***(3D)***	*2004 secs /* ***1x***	*0.009886439747*	*0.009778896185*

Επιλέξαμε τυχαία 4 παραδείγματα για να παρουσιάσουμε την βελτιστοποίηση στη μέθοδο μας. Το γεγονός ότι η GPU φαίνεται να παράγει ορθότερες εκτιμήσεις είναι τυχαίο, σε γενικές γραμμές οι εκτιμήσεις των δύο υλοποιήσεων έχουν παραπλήσιες τιμές. Ιδιαίτερο ενδιαφέρον παρουσιάζει η αυξομείωση της χρονοβελτίωσης, με αποκορύφωμα το τελευταίο από τα 4 παραδείγματα. Αν και δεν οδήγησε σε ουσιαστική διαφορά στο συνολικό χρόνο εκτέλεσης, η συνάρτηση που επιλέξαμε να επιταχύνουμε είχε χρονοβελτίωση ***10.67x*** όμως η διάρκεια εκτέλεσής της ήταν *32* και *3* δευτερόλεπτα σε επεξεργαστή και κάρτα γραφικών, αντίστοιχα. Αυτό είναι ένδειξη πως μπορούμε να λάβουμε περαιτέρω χρονοβελτίωση με την βελτιστοποίηση της συνάρτησης που επιλύει τα συστήματα που προκύπτουν (*μέθοδος conjugate gradient*) μετά τη διαδικασία εκτίμησης (***monte_carlo***).

Μέσο σφάλμα

Η επιταχυμένη υλοποίηση τείνει να παράγει καλύτερες προσεγγίσεις, δηλαδή με μικρότερο μέσο σφάλμα. Αυτό οφείλεται στο γεγονός ότι τα βάρη των μονοπατιών αθροίζονται χρησιμοποιώντας 768 double precision floating point αριθμούς. Η αρχική υλοποίηση χρησιμοποιούσε 1 τέτοια μεταβλητή. Λόγω της φύσης των floating point αριθμών όταν προστίθενται μεταξύ τους αριθμοί εισάγεται ένα σφάλμα στο τελικό αποτέλεσμα, το οποίο μεγαλώνει όσο μεγαλώνει η απόσταση των αριθμών. Επειδή πλέον χρησιμοποιούνται πολλές μεταβλητές για την άθροιση των μονοπατιών είναι λιγότερο πιθανό να προστεθούν αριθμοί πολύ διαφορετικοί σε σχέση με την πρώτη υλοποίηση που ήταν "απόλυτα σίγουρο" μιας και υπήρχε μόνο 1 μεταβλητή στην

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

οποία γινόταν όλο το άθροισμα για ένα σημείο.

Μεθοδολογία

Οι αλλαγές που κάναμε έγιναν με τέτοιο τρόπο ώστε ο τρόπος υπολογισμού της μεθόδου ***monte_carlo*** να επιλέγεται κατά την εκτέλεση της εφαρμογής. Αυτό μας οδήγησε αρχικά στην αλλαγή του τρόπου δέσμευσης μνήμης για την αποθήκευση των δεδομένων εισόδου της συνάρτησης. Από μία λίστα με λίστες μεταβλητών double precision καταλήξαμε πάλι σε μία δισδιάστατη δομή της οποίας όμως τα στοιχεία είναι αποθηκευμένα σε διαδοχικές θέσεις μνήμης. Έτσι δεν χρειάστηκαν να γίνουν αλλαγές στον αρχικό κώδικα ενώ η μεταφορά δεδομένων προς την GPU έγινε στη συνέχεια με πολύ απλό τρόπο.

Το επόμενο βήμα ήταν να δημιουργηθεί μία απλή υλοποίηση του πυρήνα της ***monte_carlo***, δηλαδή ο υπολογισμός ενός μόνο μονοπατιού, σε γλώσσα C ώστε να μην έχουμε τα overheads της C++ και να μπορεί να γίνει απεσφαλμάτωση γρήγορα και εύκολα πριν καν γίνει η τελική υλοποίηση στην κάρτα γραφικών. Σε αυτό το σημείο εντοπίσαμε πως υπήρχε μια βοηθητική συνάρτηση η οποία καλείται πολλές φορές και έκανε χρήση τεσσάρων if-statements ώστε να υπολογίσει την ελάχιστη τιμή τεσσάρων παραστάσεων. Κάναμε μερικές αλλαγές στον κώδικα ώστε να μειώσουμε τους ελέγχους έχοντας ως σκοπό να καταλήξουμε σε γρηγορότερη εκτέλεση, δυστυχώς όμως η επιτάχυνση λόγω αυτής της βελτιστοποίησης ήταν μηδαμινή και στις δύο υπολογιστικές μονάδες.

Η OpenCL υλοποίηση χρησιμοποιεί πολλαπλά νήματα για τον υπολογισμό της τιμής σε ένα σημείο. Αυτό γίνεται αναθέτοντας σε καθένα από αυτά ένα κλάσμα του συνολικού αριθμού μονοπατιών που πρέπει να "περπατηθούν" ώστε να προκύψει η εκτιμώμενη τιμή του σημείου. Ενδεικτικά αναφέρουμε πως η τρέχουσα υλοποίηση όταν εκτελείται σε μία GeForce GTX 480 χρησιμοποιεί **768** τέτοια νήματα για τον υπολογισμό κάθε σημείου, όταν ο χώρος έχει 2 διαστάσεις.

Αφού επιβεβαιώσαμε την ορθότητα του κώδικα εκτελώντας πολλαπλά πειράματα και έχοντας τις ίδιες τιμές που προκύπτουν από την αρχική υλοποίηση συνεχίσαμε με το σχεδιασμό του OpenCL Kernel. Αυτή τη φορά όμως λόγω αύξησης στον αριθμό των καταχωρητών που χρησιμοποιούνται από τον OpenCL Kernel για τα μονοπάτια σε 3D χώρο τα νήματα που χρησιμοποιούνται για την εκτίμηση ενός σημείου μειώθηκαν στα **576**.

Τέλος και στους 2 kernels χρησιμοποιήσαμε τις native υλοποιήσεις των συναρτήσεων ***cos/log/sin*** σε σημεία που δεν επηρεάζουν την ακρίβεια της εκτίμησης, μιας και εφαρμόζονται σε δεδομένα τα οποία παράγονται με ψευδοτυχαίο τρόπο. Έτσι η μείωση της ακρίβειας που υπεισέρχεται λόγω των native υλοποιήσεων εμφανίζεται ουσιαστικά ως επιπλέον τυχαιότητα, μάλιστα παρατηρήσαμε ότι σε πολλές περιπτώσεις η ακρίβεια των εκτιμήσεων βελτιώνεται.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

3. Βιβλιογραφία

[PCI] V. Vassiliadis, C.D. Antonopoulos, G. Zindros. Automating data management in heterogeneous systems using polyhedral analysis. *Proceedings of the 19th Panhellenic Conference on Informatics (PCI 2015)*. Pages 317-322. October 2015. Athens, Greece.

[ISL] S. Verdoolaege. ISL: An integer set library for the polyhedral model. In Mathematical Software (ICMS 2010), pages 299-302. Springer, 2010.

[PET] S. Verdoolaege and T. Grosser. Polyhedral extraction tool. In Proceedings of Second International Workshop on Polyhedral Compilation Techniques (IMPACT'12), 2012.

[PolyLib] D. K. Wilde. A library for doing polyhedral operations. Parallel Algorithms and Application, 15(3-4):137-166, 2000.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης