1. Οδηγίες για Αποδοτικό MPI Κώδικα Heat2D

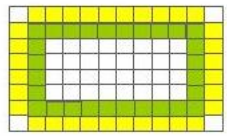
HEAT2D is based on a simplified two-dimensional heat equation domain decomposition. The initial temperature is computed to be high in the middle of the domain and zero at the boundaries. The boundaries are held at zero throughout the simulation. During the time-stepping, an array containing two domains 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.

Τρείς παράγοντες βελτίωσης απόδοσης κώδικα: Καλός σχεδιασμός (μεθοδολογία Foster) διαμοιρασμού δεδομένων (πινάκων) και επικοινωνίας, μείωση αδρανούς χρόνου, μείωση ακολουθιακών υπολογισμών.

Σχεδιασμός διαμοιρασμού δεδομένων και επικοινωνίας

Αφορά κύρια τον ισότιμο **Διαμοιρασμό Δεδομένων** (φόρτου) και **Επικοινωνίας** μεταξύ των διεργασιών και επιλογή καλής κλιμάκωσης. Η μεθοδολογία του Foster οδηγεί στον καλό σχεδιασμό, που για την εργασία σας σημαίνει

1. Διαμοιρασμός δεδομένων σε δυο διαστάσεις (block) του πίνακα. Κάθε διεργασία αναλαμβάνει ένα block, στο παρακάτω σχήμα τα πράσινα και άσπρα στοιχεία. Οι μεταπτυχιακοί εφαρμόζουν την μεθοδολογία Foster και δείχνουν ότι επιτυγχάνουμε καλύτερη κλιμάκωση, οι προπτυχιακοί απλά εφαρμόζουν τον διαμοιρασμό.
2. Τα εξωτερικά (πράσινα) σημεία σειρών (από Βορά και Νότο) και στηλών (Ανατολή και Δύση) χρειάζονται αντίστοιχα εξωτερικά στοιχεία γειτονικών διεργασιών. Αντί Send/Recv μεμονωμένων στοιχείων συμφέρει λόγω της καθυστέρησης (latency) η αποστολή/λήψη ολόκληρων σειρών και στηλών και αντίστοιχη αποθήκευση τους τοπικά σχηματίζοντας άλω (halo points). Επομένως χρειάζεται να αυξήσουμε το μέγεθος των πινάκων «πριν» και «μετά» κατά 2 στήλες και 2 γραμμές για την άλω (halo points), τα κίτρινα στοιχεία στον παρακάτω σχήμα .



Μείωση Αδρανούς (idle) Χρόνου λόγω επικοινωνίας

3. **Εφαρμόζουμε επικάλυψη επικοινωνίας με υπολογισμούς**. Έχουμε 3 ειδών στοιχεία του πίνακα. α) Εσωτερικά (λευκά), που οι τιμές της επόμενης γενεάς (time step) χρειάζονται αποκλειστικά τοπικά στοιχεία και δεν εξαρτώνται από την άλω, β) Εξωτερικά (πράσινα, πρώτη και τελευταία σειρά και αντίστοιχη στήλη) που οι τιμές της επόμενης γενεάς (time step) χρειάζονται τοπικά στοιχεία, αλλά επίσης στοιχεία της άλως που προέρχονται από τις γειτονικές διεργασίες, και γ) τα στοιχεία της άλω (κίτρινα). Αναλυτικά:
    - Αποστολές με Isend της $1^{ης}$ γραμμής στην άλω της διεργασίας στο «Βορρά» (Β), της τελευταίας γραμμής στην άλω της διεργασίας στο «Νότο» (Ν), της $1^{ης}$ στήλης στην άλω της διεργασίας της «Δύσης» (Δ) και της τελευταίας στήλης στην άλω της διεργασίας στην «Ανατολή» (Α).
    - Συμμετρικές λήψεις με Irecv των γραμμών και στηλών της άλω της διεργασίας από τις γειτονικές Β, Ν, Δ, Α.
    - Ενώ γίνεται η non-blocking μεταβίβαση γραμμών και στηλών, υπολογίζουμε τις νέες εσωτερικές τιμές που δεν εξαρτώνται από τα στοιχεία της άλω (λευκά).

- Ακολουθούν 4 Wait για την ολοκλήρωση της λήψης γραμμών και στηλών της άλω με τα αντίστοιχα Irecv.
- Κατόπιν υπολογίζουμε τα στοιχεία των πρώτων και τελευταίων γραμμών και στηλών που χρειάζονται την άλω.
- Ακολουθούν 4 Wait για την ολοκλήρωση της αποστολής γραμμών και στηλών προς την άλω με τα αντίστοιχα Isend
- Ο πίνακας «μετά» γίνεται ο «πριν»

Η δομή του σώματος της Κεντρικής επανάληψης (σταθερός αριθμός επαναλήψεων-time steps) την απόδοση της οποίας μελετάμε.

Isend (SRequest) X 4 (B,N,Δ,A)
Irecv (RRequest) X 4 (B,N,Δ,A)
Υπολογισμός νέων εσωτερικών στοιχείων (άσπρα στοιχεία στο σχήμα)
Wait (RRequest) X 4 (B,N,Δ,A) ή WaitAll (array of RRequests)
Υπολογισμός νέων Εξωτερικών στοιχείων (πράσινα στοιχεία στο σχήμα)
Wait(SRequest) X 4 (B,N,Δ,A) ή WaitAll (array of SRequests)
Πριν Πίνακας = Μετά Πίνακας

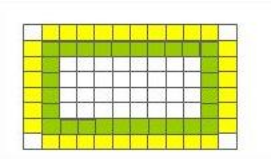
4. Με ποια διάταξη γίνονται οι αποστολές και λήψεις; Ανωτέρω αναφέρεται η σειρά Isend-Irecv. Ίσως καλύτερα να είναι η αντίστροφη Irecv-Isend, ώστε να είναι έτοιμες οι διεργασίες να δεχτούν μηνύματα.
5. Τοποθέτηση διεργασιών που επικοινωνούν στον ίδιο κόμβο, γίνεται έμμεσα μέσω τοπολογιών διεργασιών Cartesian για την μείωση κόστους επικοινωνίας. Ελέγχουμε την τοποθέτηση με MPI_Get_processor_name.
6. Χρήση datatypes στα Isend/Irecv για αποστολή/λήψη σειρών και **οπωσδήποτε** στηλών. Μπορεί να γίνει και με sub-cartesian. Αποφυγή αντιγραφών τιμών, **όχι** αντιγραφή σε buffer και μετά αποστολή, και αντίστοιχα λήψη.
7. Λόγω επικοινωνίας με σταθερούς γείτονες εφαρμόζουμε Persistent Communication, MPI_Send_init, MPI_Rsend_init, MPI_Start για να μην επαναϋπολογίζονται οι τιμές παραμέτρων envelope.
8. Υπολογισμός ranks των σταθερών γειτόνων μια φορά έξω από κεντρική επανάληψη. Προτιμάτε, όπου είναι δυνατόν οι πράξεις να γίνονται μια φορά έξω από την κεντρική επανάληψη.
9. Για ανύπαρκτους γείτονες (οριακές θέσεις διεργασιών) ορίζουμε τα αντίστοιχα ranks για αποστολή και λήψη ως MPI_PROC_NULL για να αποφύγουμε ελέγχους (if, case, κλπ) περιπτώσεων θέσεων μέσα στην κεντρική επανάληψη.

Μείωση ακολουθιακών υπολογισμών

10. Δυναμικοί Πίνακες, ούτως ώστε να κάνουμε ανταλλαγή δεικτών πίνακα «πριν» και «μετά» και **όχι** αντιγραφή τιμών πίνακα με διπλό for.
11. Μέσα στην κεντρική επανάληψη ελαττώνετε ακολουθιακούς υπολογισμούς
    a. Όχι κλήσεις συναρτήσεων, εκτός αν έχουν δηλωθεί inline
    b. Περιορισμός αναθέσεων τιμών, παραστάσεις και όχι ανάθεση σε προσωρινές μεταβλητές
    X = παράσταση (μεγάλη)
    c. Γενικά περιορισμός περιττών πράξεων, ελέγχων, κλπ. Π.χ. για το reduce (αν έχει αλλάξει κάποια τιμή) μπορεί να γίνει κατά τον υπολογισμό των νέων τιμών και όχι με επανάληψη του διπλού for, αφού οι αναθέσεις και έλεγχοι i, j στο for στοιχίζουν)
12. Compile με -O3

2. Οδηγίες για Αποδοτικό Υβριδικό Κώδικα Heat2D

Για τον υβριδικό προγραμματισμό MPI+OpenMp υπάρχουν αρκετές προσεγγίσεις. Η πιο απλή είναι να παραλληλοποιήσετε τα δυο ακολουθιακά τμήματα μέσα στην κεντρική επανάληψη α) Τον υπολογισμό των νέων εσωτερικών στοιχείων (άσπρα) που δεν εξαρτώνται από την άλω και β) Τον υπολογισμό των νέων Εξωτερικών στοιχείων μετά την λήψη της άλω (πράσινα).



Για το α) παραλληλοποιείτε ένα διπλό for και
για το β) παραλληλοποιείτε τα 4 for. Ίσως να έχετε μεγαλύτερη ευελιξία, όταν όλα τα νήματα παραλληλοποιούν τους υπολογισμούς της ίδιας στήλης ή σειράς (data παραλληλισμός) και όχι διαφορετικά νήματα να παραλληλοποιούν μια στήλη ή σειρά (domain παραλληλισμός).

Επίσης η υλοποίηση για την διαπίστωση μη αλλαγής των τιμών των πινάκων «πριν» και «μετά» μπορεί να γίνει είτε σε ξεχωριστή επανάληψη (που δεν συνιστάται), είτε ενσωματωμένη στις προηγούμενες επαναλήψεις α) και β)). Πάντως και στις δυο περιπτώσεις θα χρειαστείτε openmreduce μεταξύ των νημάτων στην ίδια διεργασία και κατόπιν το MPI reduce διεργασιών.

Τέλος χρειάζεται προσοχή η θέση που δημιουργούνται τα νήματα. Αν δημιουργούνται μέσα στην επανάληψη στος 2 ή 3 θέσεις έχουμε το επιπλέον κόστος δημιουργίας και καταστροφής τους σε κάθε επανάληψη. Καλύτερο θα είναι να δημιουργηθούν μια φορά έξω από την επανάληψη, όπως το τελευταίο παράδειγμα του Pacheco για την ταξινόμηση στοιχείων με την εντολή

```
#pragma omp parallel
```

και μέσα στην επανάληψη να αναθέσετε επαναλήψεις για α) και β) στα νήματα με την εντολή

```
#pragma omp parallel for schedule(scheduling-type)
```

Δοκιμάστε διάφορους τρόπους scheduling. Το static μάλλον θα είναι καλύτερο με chunk_size=? (πειραματισμός) και διαφορετικό αριθμών νημάτων (2,4,8). Δοκιμάστε επίσης το διπλό for sto α) με collapse(2).

IOANNIS KOTRONIS OR COTRONIS

Digitally signed by IOANNIS KOTRONIS OR COTRONIS
Date: 2018.12.10 15:53:54 +02'00'