Οδηγίες για Αποδοτικό MPI Κώδικα Conway's Game of Life

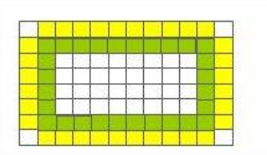
Χρησιμοποιούμε 2 πίνακες (char) «πριν» και «μετά». Ο «μετά» υπολογίζεται από τις τιμές του «πριν» βάσει των 4 κανόνων. Για την επόμενη γενεά ο «μετά» γίνεται «πριν». Οι δυο πίνακες διαμερίζονται στις διεργασίες.

Τρείς παράγοντες βελτίωσης απόδοσης κώδικα (11 σημεία): Καλός σχεδιασμός (μεθοδολογία Foster) διαμοιρασμού δεδομένων (των 2 πινάκων) και επικοινωνίας (σημεία 1, 2), μείωση αδρανούς χρόνου (σημεία 3,4,5,6,7 και 8) και μείωση ακολουθιακών υπολογισμών (9,10,11).

Σχεδιασμός διαμοιρασμού δεδομένων και επικοινωνίας (Foster)

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 σε blocks.
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. Η επέκταση καλύπτει και τα 4 γωνιακά.



Μείωση Αδρανούς (idle) Χρόνου και Overhead

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 (non-blocking) 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). Επίσης αποστολή πράσινων γωνιακών στοιχείων στις διεργασίες ΒΑ, ΒΔ, ΝΑ, ΝΔ
    - Συμμετρικές λήψεις με Irecv (non-blocking) των γραμμών και στηλών της άλω της διεργασίας (κίτρινα) από τις γειτονικές Β, Ν, Δ, Α και γωνιακά σημεία.
    - Ενώ γίνεται η non-blocking μεταβίβαση γραμμών, στηλών και γωνιακών στοιχείων, υπολογίζουμε τις νέες εσωτερικές τιμές που δεν εξαρτώνται από τα στοιχεία της άλω (λευκά).
    - Ακολουθούν 8 Wait για την ολοκλήρωση της λήψης γραμμών, στηλών και γωνιακών της άλω με τα αντίστοιχα Irecv.

- Κατόπιν υπολογίζουμε τα στοιχεία των πρώτων και τελευταίων γραμμών και στηλών που χρειάζονται την άλω.
- Ακολουθούν 8 Wait για την ολοκλήρωση της αποστολής γραμμών, στηλών προς και γωνιακών την άλω με τα αντίστοιχα Isend
- Ο πίνακας «μετά» γίνεται ο «πριν». Χρησιμοποιούμε δείκτες και ΠΟΤΕ αντιγραφή!

Τα ανωτέρω εκφράζονται στην δομή της Κεντρικής επανάληψης (σταθερός αριθμός επαναλήψεων-time steps) την απόδοση της οποίας μελετάμε.

MPI_Barrier (συγχρονισμός διεργασιών πριν τις μετρήσεις)
Start MPI_Wtime
For #επαναλήψεων (σταθερός σε όλες τις μετρήσεις σας)
Irecv (RRequest) X 8 (B,N,Δ,A + γωνιακά)
Isend (SRequest) X 8 (B,N,Δ,A + γωνιακά)
Υπολογισμός εσωτερικών στοιχείων «μετά» (άσπρα στοιχεία στο σχήμα) και
ένδειξη κενού πίνακα ή μη αλλαγής (διπλό for)
Wait (RRequest) X 8 (B,N,Δ,A+γωνιακά) ή WaitAll (array of RRequests)
Υπολογισμός εξωτερικών στοιχείων «μετά» (πράσινα στο σχήμα)-4 for
Πριν Πίνακας = Μετά Πίνακας
(reduce για έλεγχο εδώ)
Wait(SRequest) X 8 (B,N,Δ,A+γωνιακά) ή WaitAll (array of SRequests)
End for
End MPI_Wtime

4. Οι εντολές Recv πριν τις Send (πρώτα Irecv μετά Isend), ώστε να είναι έτοιμες οι διεργασίες να δεχτούν μηνύματα.
5. 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.
6. Υπολογισμός ranks των 8 *σταθερών* γειτόνων μια φορά έξω από κεντρική επανάληψη. Προτιμάτε, όπου είναι δυνατόν οι πράξεις να γίνονται μια φορά έξω από την κεντρική επανάληψη.
7. Τοποθέτηση διεργασιών που επικοινωνούν στον ίδιο κόμβο. Γίνεται έμμεσα μέσω τοπολογιών διεργασιών Cartesian για την μείωση κόστους επικοινωνίας. Ελέγχουμε την τοποθέτηση με το mpiP. Επίσης μέσω τοπολογιών βρίσκουμε τους 8 γείτονες.
8. Λόγω επικοινωνίας με σταθερούς γείτονες εφαρμόζουμε Persistent Communication, MPI_Send_init, MPI_Start για να μην επαναϋπολογίζονται οι τιμές παραμέτρων των κλήσεων Isend, Irecv. Προσοχή, χρειάζονται 2 οικογένειες MPI_Send_init, MPI_Start ανάλογα με την επανάληψη.

Μείωση ακολουθιακών υπολογισμών στην κεντρική επανάληψη

9. Δυναμικοί Πίνακες ή δείκτες, ούτως ώστε να κάνουμε απόδοση της τιμής δείκτη στον πίνακα «πριν» από τον «μετά» αποφεύγοντας την αντιγραφή τιμών πίνακα με διπλό for.
10. Μέσα στην κεντρική επανάληψη εν γένει ελαττώνετε ακολουθιακούς υπολογισμούς
    a. Όχι κλήσεις συναρτήσεων, εκτός αν έχουν δηλωθεί inline
    b. Περιορισμός αναθέσεων τιμών, προτιμότερο οι μεγάλες παραστάσεις από την ανάθεση σε προσωρινές μεταβλητές (εντολή MAD ως μια)

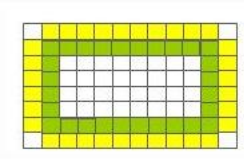
        X = παράσταση (μεγάλη)

c. Γενικά περιορισμός αναθέσεων, ελέγχων, πράξεων, κλπ. Π.χ. η τιμή για το reduce (αν δηλαδή έχει αλλάξει κάποια τιμή) μπορεί να γίνει κατά τον υπολογισμό των νέων τιμών με έναν απλό έλεγχο και όχι με νέα επανάληψη διπλού for, (οι αναθέσεις και έλεγχοι i, j στα for στοιχίζουν)

11. Μετρήσεις με Compile με -O3 (αρχικά με -O0 για debug)

Οδηγίες για Αποδοτικό Υβριδικό Κώδικα

12. 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 (πράσινα) μετά την λήψη των δυο σειρών και δυο στηλών της άλω (κίτρινα)



Για το α) παραλληλοποιείτε ένα διπλό for (δοκιμάστε block sizes, collapse) 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 και όχι διαφορετικά νήματα να παραλληλοποιούν μια στήλη ή σειρά.

13. Βρείτε πρώτα σε έναν μόνο κόμβο (8 πυρήνες) τον βέλτιστο συνδυασμό αριθμού διεργασιών (δ) και νημάτων (ν), π,χ. να εξετάσετε συνδυασμούς 1δ-8ν, 1δ-16ν, 2δ-4ν. 2δ-8ν, 4δ-2ν, 4δ-4ν. Δοκιμάστε διάφορους τρόπους scheduling. Το static μάλλον θα είναι καλύτερο (chunk_size=? πειραματισμός). Δοκιμάστε επίσης το διπλό for στο α) με collapse(2).

14. Κατόπιν να επιλέξετε τον καλύτερο συνδυασμό αριθμού δ-ν και scheduling και με αυτόν συνεχίστε την μελέτη κλιμάκωσης δεδομένων και επεξεργαστών δημιουργώντας διεργασίες για 2, 4, 8 και 10 κόμβους.

15. 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 openmp-reduce μεταξύ των νημάτων στην ίδια διεργασία και κατόπιν το MPI reduce διεργασιών. (Σχετίζεται με το 10c)

16. Τέλος χρειάζεται προσοχή που δημιουργούνται τα νήματα. Αν δημιουργούνται μέσα στην κεντρική επανάληψη (πριν το διπλό for) 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 κεντρική επανάληψη να αναθέσετε επαναλήψεις για α) και β) στα νήματα με την εντολή

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

Προσοχή, όμως οι εντολές MPI θα πρέπει να εκτελούνται μόνο από το MASTER νήμα!

17. Υπάρχουν και άλλες προσεγγίσεις για χρήση νημάτων, για τους πιο θαρραλέους., π.χ. Μια διαφορετική υλοποίηση επικάλυψης επικοινωνίας με υπολογισμούς είναι ένα νήμα να εκτελεί blocking Send+Recv (ή ακόμη καλύτερα SendRecv) και τα υπόλοιπα να εκτελούν το διπλό for. Μετά χρειάζεται omp barrier.

18. Επίσης στις σημειώσεις που έχω δώσει θα βρείτε και άλλες προσεγγίσεις υβριδικών προγραμμάτων.