Οδηγίες για Αποδοτικό MPI και Υβδριδικό MPI+OpenMp Κώδικες για την Εργασία 2020-21Jacobi (22/12/2020)

Έχουμε τρείς κατηγορίες οδηγιών βελτίωσης απόδοσης κώδικα (18 σημεία):Ξεκινήστε με α) μείωση ακολουθιακού χρόνου (σημεία 1,2,3). Συνεχίστε με β) Σχεδιασμό (μεθοδολογία Foster) διαμοιρασμού δεδομένων των 2 πινάκων και επικοινωνίας (σημεία 4,5) γ) μείωση αδρανούς χρόνου και επιπλέον χρόνου-overhead (σημεία 6 έως 11). Για το υβριδικό πρόγραμμα ακολουθήστε τις οδηγίες (σημεία 12 έως 17).

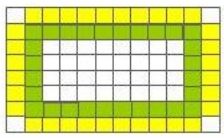
Α. Μείωση ακολουθιακού χρόνου στην κεντρική επανάληψη

1. 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. (ήδη υλοποιημένο)
2. Μέσα στην κεντρική επανάληψη ελαττώνετε κόστος ακολουθιακών υπολογισμών
    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), π.χ. X = παράσταση (μεγάλη)
    c. Γενικά περιορισμός αναθέσεων, ελέγχων, πράξεων, περιττός επαναϋπολογισμός, κλπ.
3. Επιβεβαιώστε με μετρήσεις με Compile με -O3 (αρχικά με -O0 για debug και έλεγχο αποτελεσμάτων)

Β. Σχεδιασμός διαμοιρασμού δεδομένων και επικοινωνίας (στάδια 1,2,3 Foster) για το MPI πρόγραμμα

Αφορά κύρια τον ισότιμο **Διαμοιρασμό Δεδομένων** (φόρτου) και **Επικοινωνίας** μεταξύ των διεργασιών και επιλογή καλής κλιμάκωσης. Χρησιμοποιούμε 2 πίνακες (double) u_old («πριν») και u («μετά»). Ο u υπολογίζεται από τις τιμές του u_old βάσει του τύπου-stencil. Για την επόμενη γενεά ο u «μετά» γίνεται u_old «πριν». Οι δυο πίνακες διαμερίζονται στις mpi διεργασίες. Η μεθοδολογία του Foster οδηγεί στον καλό σχεδιασμό, που για την εργασία σας σημαίνει

4. Διαμοιρασμός δεδομένων σε δυο διαστάσεις (block-block) του πίνακα. Κάθε διεργασία αναλαμβάνει ένα block, στο παρακάτω σχήμα τα πράσινα και άσπρα στοιχεία.



5. 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 (datatype) και αντίστοιχη αποθήκευση τους τοπικά σχηματίζοντας άλω (halo points). Επομένως χρειάζεται να αυξήσουμε το μέγεθος των πινάκων u_old «πριν» και u «μετά» κατά 2 στήλες και 2 γραμμές για την άλω (halo points), τα κίτρινα στοιχεία στον παρακάτω σχήμα.

Γ. Μείωση Αδρανούς (idle) και Επιπλέον (overhead) Χρόνου

6. **ΟΠΩΣΔΗΠΟΤΕ** χρήση datatypes για αποστολή/λήψη σειρών και **οπωσδήποτε** στηλών. Αποφυγή αντιγραφών τιμών, **όχι** αντιγραφή σε buffer και μετά αποστολή, και αντίστοιχα λήψη. Μπορεί να γίνει και με sub-cartesian
7. **Εφαρμόζουμε επικάλυψη επικοινωνίας με υπολογισμούς**. Έχουμε 3 είδη στοιχείων 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 αντίστοιχα στήλες) 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) που λαμβάνονται από γειτονικές διεργασίες. Επομένως:

- Αποστολές με Isend (non-blocking) της 1ης γραμμής στην άλω της διεργασίας στο «Βορρά» (Β), της τελευταίας γραμμής στην άλω της διεργασίας στο «Νότο» (Ν), της 1ης στήλης στην άλω της διεργασίας της «Δύσης» (Δ) και της τελευταίας στήλης στην άλω της διεργασίας στην «Ανατολή» (Α). (πράσινα)
- Συμμετρικές λήψεις με Irecv (non-blocking) των γραμμών και στηλών της άλω της διεργασίας (κίτρινα) από τις γειτονικές Β, Ν, Δ, Α. (κίτρινα)
- 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 (ή WaitAll) 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». 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
Κεντρικό For #επαναλήψεων (σταθερός σε όλες τις μετρήσεις σας)
   Irecv (RRequest) X 4 (Β,Ν,Δ,Α)
   Isend (SRequest) X 4 (Β,Ν,Δ,Α)
         Υπολογισμός εσωτερικών στοιχείων «μετά» (άσπρα) και υπολογισμός
         error (διπλό for)
   Wait (RRequest) X 4 (Β,Ν,Δ,Α) ή WaitAll (array of RRequests)
         Υπολογισμός εξωτερικών στοιχείων «μετά» (πράσινα) και υπολογισμός
         error -4 for για 2 γραμμές και 2 στήλες
         Πριν Πίνακας u = Μετά Πίνακας u_old
         υπολογισμός σύγκλισης error με reduce
   Wait(SRequest) X 4 (Β,Ν,Δ,Α) ή WaitAll (array of SRequests)
End Κεντρικό for
End MPI_Wtime
```

8. Οι εντολές Recv πριν τις Send (πρώτα Irecv μετά Isend), ώστε να είναι έτοιμες οι διεργασίες να δεχτούν μηνύματα. Μπορείτε να δοκιμάσετε και Irsend.
9. Υπολογισμός ranks των 4 *σταθερών* γειτόνων μια φορά έξω από κεντρική επανάληψη. Γενικά, προτιμάτε, όπου είναι δυνατόν οι πράξεις να γίνονται μια φορά έξω από την κεντρική επανάληψη.
10. 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 (u, u_old).

11. Τοποθέτηση διεργασιών που επικοινωνούν στον ίδιο κόμβο. Γίνεται έμμεσα μέσω τοπολογιών διεργασιών Cartesian (με rank reorder) για την μείωση κόστους επικοινωνίας. Ελέγχουμε την τοποθέτηση με το mpiP. Επίσης μέσω τοπολογιών βρίσκουμε τους 4 γείτονες.

Οδηγίες για Αποδοτικό Υβριδικό MPI+OpenMp Κώδικα

12. Για τον υβριδικό προγραμματισμό MPI+OpenMp υπάρχουν αρκετές προσεγγίσεις. Η πιο απλή είναι να παραλληλοποιήσετε τα δυο ακολουθιακά τμήματα (for επαναλήψεις) 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 το 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 και όχι διαφορετικά νήματα να παραλληλοποιούν μια στήλη ή σειρά.
13. Αξιολογήστε πρώτα σε έναν μόνο argo κόμβο (8 πυρήνες) τον βέλτιστο συνδυασμό αριθμού διεργασιών (δ) και νημάτων (ν), π,χ. να εξετάσετε συνδυασμούς 1δ-8ν, 2δ-4ν, 2δ-8ν, 4δ-2ν, 4δ-4ν. Τα πιο πιθανά είναι τα 2δ-4ν και 2δ-4ν. Δοκιμάστε διάφορους τρόπους for 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 σε έναν κόμβο και συνεχίστε την μελέτη κλιμάκωσης δεδομένων και επεξεργαστών δημιουργώντας διεργασίες σε περισσότερους 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.
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 (πιο εύκολο). 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 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. Επίσης στις σημειώσεις που έχω δώσει θα βρείτε και άλλες προσεγγίσεις υβριδικών προγραμμάτων.