Εργασία 2019-20: Conway's Game of Life
Γενικές Οδηγίες για την Εργασία MPI-OpenMp-Cuda

Ζητείται

Να σχεδιάσετε, να υλοποιήσετε και να αξιολογήσετε παράλληλα προγράμματα του **Conway's Game of Life** σε περιβάλλοντα MPI, υβριδικό MPI+OpenMp και Cuda. Παραδίδετε α) τεκμηρίωση με τον σχεδιασμό και την αξιολόγηση και β) κώδικα για τις 3 υλοποιήσεις.

Ιστορία-Κανόνες παιχνιδιού

Το Παιχνίδι της Ζωής αναπτύχθηκε το 1970 από τον John Conway στο Πανεπιστήμιο του Καίμπριτζ. Το παιχνίδι καταδεικνύει ότι μερικοί απλοί τοπικοί κανόνες μπορούν να οδηγήσουν σε ενδιαφέρουσα μεγάλης κλίμακας συμπεριφορά ζωής (γέννηση, αναπαραγωγή, θάνατος). Παίζεται σε ένα περιοδικό δυσδιάστατο πλέγμα (NxN) από κελιά ή κύτταρα, τα οποία μπορούν να βρίσκονται σε μια από δυο καταστάσεις: ζωντανά ή νεκρά. Το παιχνίδι δεν έχει παίκτες, δηλαδή δεν απαιτεί εισαγωγή δεδομένων κατά την εξέλιξή του, αλλά αυτή εξαρτάται αποκλειστικά από το αρχικό σχέδιο του πλέγματος Κάθε κελί θεωρείται πως έχει οκτώ γείτονες, οι οποίοι επηρεάζουν την κατάστασή του. Το σύνολο των ζωντανών κελιών αποτελεί τον πληθυσμό του πλέγματος. Η εξέλιξη γίνεται σε διακριτά βήματα, τις γενεές, και η ανανέωση του πλέγματος από γενεά σε γενεά γίνεται «ταυτόχρονα», δηλαδή η κατάσταση κάθε κελιού στην επόμενη γενιά εξαρτάται αποκλειστικά από την κατάσταση του ιδίου και των οκτώ γειτόνων του στη παρούσα γενιά, βάση ορισμένων κανόνων.

Το παιχνίδι αρχίζει είτε με μια τυχαία επιλογή των κατειλημμένων θέσεων είτε με ένα σχέδιο που διαβάζεται από ένα αρχείο. Από αυτήν την αρχική γενεά, η επόμενη γενεά υπολογίζεται χρησιμοποιώντας τους ακόλουθους κανόνες:

1. Εάν ένας οργανισμός (κατειλημμένη θέση) έχει 0 ή 1 γειτονικούς οργανισμούς, ο οργανισμός πεθαίνει από μοναξιά.
2. Εάν ένας οργανισμός έχει 2 ή 3 γειτονικούς οργανισμούς, ο οργανισμός επιζεί στην επόμενη γενεά.
3. Εάν ένας οργανισμός έχει 4 έως 8 γειτονικούς οργανισμούς, ο οργανισμός πεθαίνει λόγω υπερπληθυσμού.
4. Εάν μία μη κατειλημμένη θέση έχει ακριβώς 3 γειτονικούς οργανισμούς, αυτή η θέση θα καταληφθεί στην επόμενη γενεά από έναν νέο οργανισμό, δηλαδή ένας οργανισμός γεννιέται

https://www.conwaylife.com/
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Τεκμηρίωση: Περιεχόμενο και Δομή

1. Εισαγωγή
2. Σχεδιασμός Διαμοιρασμού Δεδομένων (διαμοιρασμός σε Block, όχι σειρές, επικοινωνία, τοπολογία διεργασιών, κλπ.) ***
3. Σχεδιασμός MPI κώδικα με στόχο την μείωση αδρανούς χρόνου ή περιττών υπολογισμών (επιλογές επικοινωνίας, επικάλυψη επικοινωνίας με υπολογισμούς, αποφυγή πολλαπλών αντιγραφών με χρήση datatypes, κλπ). Έλεγχος για μη αλλαγή πλέγματος κάθε ν επαναλήψεις. Δες οδηγίες Σχεδιασμού και Ανάπτυξης που σας έχω δώσει για βελτιστοποίηση κώδικα. Να αναφέρετε *ρητά* ποιες βελτιώσεις έχετε εφαρμόσει και ποιες όχι. Επίσης αν έχετε κάνει άλλες που δεν έχω αναφέρει.
4. Μετρήσεις χρόνων εκτέλεσης, υπολογισμός speedup, efficiency παρουσίαση αποτελεσμάτων (με έλεγχο τερματισμού και χωρίς). Μελέτη κλιμάκωσης δεδομένων και επεξεργαστών. Σταθερός αριθμός επαναλήψεων για όλες τις μετρήσεις, ώστε να παίρνετε μετρήσιμους χρόνους, π.χ ακολουθιακό πρόγραμμα όχι πάνω από ~20 δευτερόλεπτα. Χρήση mpiP για τον εντοπισμό καθυστερήσεων και την βελτιστοποίηση του κώδικα MPI. Μετρήσεις για
    a. MPI πρόγραμμα μόνο με τοπική επικοινωνία, χωρίς allreduce για έλεγχο κενού πλέγματος ή μη αλλαγής πλέγματος.
    b. MPI πρόγραμμα με επιπλέον έλεγχο κενού πλέγματος ή μη αλλαγής. Προσοχή, ακόμη και να διαπιστώσετε συνθήκη τέλους να εκτελείτε όλες τις επαναλήψεις γενεών. Allreduce κάθε 10 επαναλήψεις.
    c. Να ενσωματωθούν στο MPI+allreduce εντολές OpenMp για παραλληλοποίηση υπολογισμών (π.χ εσωτερικών στοιχείων), ώστε να αναπτυχθεί υβριδικό πρόγραμμα. Συγκρίσεις με καθαρό MPI.
5. Σχεδιασμός, αξιολόγηση καθαρού προγράμματος cuda με ίδιους υπολογισμούς.
6. Συμπεράσματα
7. Επίσης να παραδώσετε την έξοδο του mpiP για τις καλύτερες σας επιδόσεις (μεγάλα δεδομένα, πολλές διεργασίες)

- *** Για το ΠΜΣ + ΗΑ επιπλέον στο 2, εφαρμογή μεθοδολογίας Foster: ανάλυση διαμερισμού με σειρές και block-block Να δείξετε ότι με block έχουμε καλύτερη κλιμάκωση. Για τις σταθερές latency και bandwidth και άλλες μετρικές του συστήματος να χρησιμοποιήσετε το mpptest. Στο 4d σύγκριση αναλυτικών υπολογισμών και πραγματικών μετρήσεων

Παράδοση

Παράδοση (αυστηρές προθεσμίες-χωρίς δυνατότητα παράτασης):
Κυριακή 16/02/2020 για εξεταστική Ιανουαρίου (διαθεσιμότητα μηχανών Marie 1-14/2) ή
Κυριακή 11/10/2020 για εξεταστική Σεπτεμβρίου (διαθεσιμότητα μηχανών Marie 26/9-09/10)
Παραδίδετε τεκμηρίωση και α) τον κώδικα MPI+allreduce+OpenMp (συνιστώμενο, αντί για 3 κώδικες, εφόσον έχετε κάνει μικρές τοπικές επεκτάσεις) και β) για CUDA (εκτός αν δεν είναι εφικτό-δες πιο κάτω οδηγίες-2).

Αξιολόγηση

Η εργασία συνολικά συνεισφέρει στον τελικό βαθμό 60%, τα γραπτά 30% και η άσκηση 10%.
Η αξιολόγηση της εργασίας με κατανομή στα 100, ως εξής

Τεκμηρίωση (Κείμενο) Παρουσίαση-Αποτελέσματα,		**(60/100)**
Συνολική εικόνα (δομή, πληρότητα, κλπ)	05/100	
Επιμερισμός τεκμηρίωσης ανά πρόγραμμα		
MPI	35/100	
MPI + allreduce	05/100	
MPI + allreduce + openMp	10/100	
Cuda	05/100	
Κώδικας		**(40/100)**
MPI	20/100	
MPI + allreduce	05/100	
MPI + allreduce +openMp	05/100	
Cuda	10/100	

Διαδικαστικά – Οδηγίες

1. Επιτρέπεται και μάλιστα ενδείκνυται η συνεργασία 2 ατόμων. Ομάδες των 3ων επιτρέπεται, όμως στην περίπτωση αυτή η εργασία θα αξιολογηθεί με πιο αυστηρά κριτήρια και θα πρέπει να περιλαμβάνει κάποιες επεκτάσεις, που συμπεριλάβετε στην τεκμηρίωση και στον κώδικα, όπως
   - Parallel I/O .Να κάνετε ξεχωριστές μετρήσεις για το I/O, με τον γνωστό τρόπο Wtime πριν και μετά το I/O. Επιλογή εξόδου μετά από κάθε βήμα.
   - Εμφάνιση αλληλουχίας γενεών στην οθόνη (visualization) με gnuplot ή άλλα εργαλεία
   - (επέκταση για ΠΠΣ) Μεθοδολογία Foster, μετρήσεις με Mpptest
   - Μεθοδολογία POP
   - Μετρήσεις στο Marie για πάνω από 80 έως 160 πυρήνες (θα τις θεωρήσω επιπλέον δουλειά για ομάδες των 3ών, αρκεί βέβαια να δείχνουν κλιμάκωση. Αν είναι περιττές, δες πιο κάτω στο 8, θα είναι αρνητικό).
   - Σύγκριση OpenMp vs. MPI Shared memory access σε έναν κόμβο
   - Πειράματα με διαφορετικά υβριδικά προγράμματα (π.χ. με νήματα που επικοινωνούν με MPI).
   - Πειράματα με Thread affinity, caching
2. Θα προσπαθήσω να έχετε διαθέσιμη nVidia κάρτα για CUDA. Αν δεν γίνει και δεν έχετε πρόσβαση σε nVidia GPU να κάνετε εναλλακτική εργασία με ένα από τα ανωτέρω. Οι ομάδες των $3^{ων}$ φυσικά κάτι επιπλέον Χ2.
3. Μπορείτε να κάνετε τον έλεγχο, βελτιώσεις ακόμη και τις τελικές μετρήσεις στο νέο cluster ARGO. Σας το προτείνω, ώστε να αποφύγετε προβλήματα συνωστισμού "τελευταίας στιγμής" είτε στο Marie είτε στο Argo. Με την δυνατότητα πρόσβασης στην ΑΡΓΩ δεν θα δεχτώ κανένα παράπονο για καθυστερήσεις και αίτημα παράτασης.
4. Να τρέχετε πάντα με ενεργοποιημένο το mpiP, να βάλετε όμως εντολές για ενεργοποίηση του μόνο στο κεντρικό for, εκεί δηλαδή που έχετε βάλει WTime και όχι για όλο το πρόγραμμα. Οδηγίες θα βρείτε στο κεφ. "Controlling the Scope of mpiP Profiling in your Application" των σημειώσεων mpiP. (όπως στην εργαστηριακή άσκηση)
5. Αν δεν παίρνετε καλά αποτελέσματα να εξετάσετε το output του mpiP. Θα σας βοηθήσει να δείτε τι δεν πάει καλά. Δείτε π.χ. που ξοδεύει χρόνο το πρόγραμμά σας (waits, sends, recvs). Μερικά προβλήματα είναι απλά, π.χ. τα waits σε λάθος θέση.
6. Μελέτη κλιμάκωσης για 1, 2,4,8, 16, 32, 64 και αν κλιμακώνει έως τις 80 διεργασίες και (βολικά) μεγέθη NXN πίνακα, ώστε να διαιρούνται ακριβώς. Ξεκινήστε με 320Χ320 με συνεχίστε με διπλασιασμό κάθε πλευράς, μέχρις ότου δεν έχουμε επιτάχυνση ή όταν ο πίνακας δεν χωράει στην μνήμη.
7. Για το υβριδικό MPI+OpenMp, 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. Κατόπιν να επιλέξετε τον καλύτερο συνδυασμό δ-ν στον ένα κόμβο και συνεχίστε την μελέτη κλιμάκωσης δεδομένων και επεξεργαστών για 2,4, 8 και 10 κόμβους.
8. Όμως, να μην κάνετε μετρήσεις για τις μετρήσεις. Αν δεν έχετε επιτάχυνση >1 μετά από κάποιον αριθμό πυρήνων να μην κάνετε τυφλά τρεξίματα με περισσότερους πυρήνες για το ίδιο μέγεθος προβλήματος. Είναι περιττές και σπατάλη πόρων.