Άντωνιαδης Σταύρος ΑΕΜ:8279

Αναφορά

Παράλληλα και Διανεμημένα Συστήματα

Άσκηση 2

Σκοπός της άσκησης ήταν η παραλληλοποίηση του προγράμματος που προσομοιώνει το Game of Life με OpenMP και Mpi. Το Game of life είναι ένα κυψελικό αυτόματο του οποίου η εξέλιξη εξαρτάται μόνο από τις αρχικές συνθήκες. Η υλοποίηση στο κομμάτι της Mpi έγινε με δύο τρόπους ένας τρόπος ήταν με Isend/Irecv και ο άλλος χρησιμοποιώντας one-sided Communication με MPI_Win_create και MPI_Get. Θα αναφέρουμε τώρα τις αλλαγές/προσθήκες που κάναμε στο δοθέν πρόγραμμα και τι επίδραση έχουν.

- **Main:** Οι αλλαγές που έγιναν εδώ ήταν να βάλουμε να εισάγεται σαν είσοδος και οι στήλες του Board με την μεταβλητή M. Στη συνέχεια χρησιμοποιήσαμε κάποιες εντολές Mpi τις MPI_Init, MPI_Comm_size, MPI_Comm_rank. Η πρώτη περνάει τα ορίσματα που έχουμε πάρει από τον χρήστη σε όλα τα processes, η δεύτερη μας επιστρέφει τον αριθμό των Tasks που έχουμε και η τελευταία μας επιστρέφει το rank/id του

process που βρισκόμαστε. Στη συνέχεια του προγράμματος αυτό που ουσιαστικά κάνουμε είναι να ελέγχουμε τον αριθμό των Tasks που έχουμε και αναλόγως να σπάμε το Board του Game of Life σε ισόποσα μέρη και να καλούμε τις συναρτήσεις για να αρχίσει η προσομοίωση του Game of Life. Έδω να πούμε ότι έχουμε βάλει να δημιουργείται ένα Task/Node. Τέλος, μετά το πέρας της υλοποίησης χρησιμοποιούμε τις MPI_Barrier,MPI_Finalize. Η πρώτη σταματάει το πρόγραμμα σε εκείνο το σημείο μέχρι να επιστρέψουν όλα τα processes, όταν όλα φτάσουν εκεί συνεχίζει στην επόμενη γραμμή του κώδικα. Η δεύτερη απλά τερματίζει την Mpi.

- **Init:** Στην συνάρτηση initialize απλά προσθέσαμε την εντολή #pragma omp parallel for για να σπάσουμε το loop σε όλα τα διαθέσιμα threads. Στην generate table αλλάξαμε την rand() με μία thread safe συνάρτηση την drand48_r και θέσαμε το seed να εξαρτάται από το id του processor και το id του Task με τέτοιο τρόπο ώστε να μην υπάρχει επικάλυψη των αριθμών που παράγει το κάθε thread σε κάθε node βλέπε και κώδικα (υπάρχουν και κάποια σχόλια). Το

generate του Board έγινε από όλους τους πυρήνες του κάθε Node μίας και χρησιμοποιήσαμε την #pragma omp parallel και #pragma omp for με reduction για το counter.

- **Play:** Οι αλλαγές που έγιναν εδώ ήταν να βάλουμε πάλι τις εντολές MPI_Comm_size, MPI_Comm_rank και αναλόγως με τον αριθμό των Tasks να κάνουμε τα κατάλληλα MPI_Isend και MPI_Irecv από το ένα process στο άλλο. Δημιουργήσαμε ένα νέο τύπο δεδομένων για νά στέλνουμε την πρώτη και την τελευταία γραμμή κάθε υποBoard για να κάνουμε σωστά την προσομοίωση χρησιμοποιώντας κυκλικές οριακές συνθήκες. Ακόμα, κάναμε latency hiding υπολογίζοντας πρώτο όλο τον πίνακα και μετά την πρώτη και την τελευταία γραμμή.

  Η υλοποίηση με one-sided Communication ήταν ακριβώς η ίδια με την διαφορά ότι στην play αντί για Isend/Irecv χρησιμοποιήσαμε τις MPI_Win_create και MPI_Get. Η πρώτη δημιουργεί ένα παράθυρο όπως λέγεται το οποίο είναι μία περιοχή της μνήμης του process που καλεί την εντολή το οποίο είναι προσβάσιμο απο όλα τα άλλα process ώστε

να μπορούν να π’ανε να διαβάσουν και να γράψουν σε αυτήν. Η δεύτερη ουσιαστικά πάει και διαβάζει από αυτή τη μνήμη και την μεταφέρει σε ένα buffer στο process που την κάλεσε.

Σκοπός της εργασίας ήταν στην προσομοίωση του Game of life, καθώς ανεβαίνει ο αριθμός των Tasks και το μέγεθος του πίνακα ο χρόνος να παραμένει σταθερός. Αυτό βλέπουμε οτι επιτυγχάνεται:

- Για μέγεθος 40000X40000 και 1 Task έκανε χρόνο 63sec.
- Για μέγεθος 80000X40000 και 2 Tasks έκανε χρόνο 64sec.
- Για μέγεθος 80000X80000 και 4 Tasks έκανε χρόνο 64sec.

Παρόμοια είναι τα αποτελέσματα με την μέθοδο one-sided Communication. Λεπτομερέστερα αποτελέσματα για το χρόνο κάθε συνάρτησης καθώς και για τους παραπάνω χρονους και για τις δύο υλοποιήσεις παρατίθενται.