Παράλληλα και Διανεμημένα Συστήματα

Εργασία 2η

Δημήτρης Παππάς

ΑΕΜ: 8391

Κατασκευή Αλγορίθμου knn:

Για την διαχείριση των δεδομένων μου όρισα ένα struct, το οποίο περιέχει εναν double για την αποθήκευση της απόστασης μεταξύ δύο σημείων κι έναν integer για την αποθήκευση του στοιχείου, με το οποίο γίνεται η σύγκριση.

```
typedef struct {
    double distance;
    int element;
}knn_struct;
```

Στη συνέχεια, χρησιμοποιώ δύο file pointers fpi, fpj, ώστε να διαβάζω δύο διαφορετικα σημεία ταυτόχρονα και να τα αποθηκεύω σε δύο μονοδιάστατους πίνακες 1x30. Επέλεξα αυτή την υλοποίηση προκειμένου ο κώδικας μου να καταλαμβάνει ελάχιστο χώρο στη μνήμη, παρόλο που τον καθηστα πιο αργό. Έτσι, είχα την δυνατότητα να τρέξω ολόκληρο το αρχείο σε ένα συστημα.

Σε περίπτωση που οι file pointer κοιτανε στο ίδιο στοιχείο (ίδια σειρά), διαβάζω το αρχείο (για να μην χάσει την σειρά του ο file pointer) υπολογίζω και αναμένω μηδενική απόσταση και με την continue ξαναπηγαίνω στη for j, ώστε να το διαβάσω το επόμενο.

Όταν η απόσταση είναι μη μηδενική τοποθετώ στο τελευταιο κελί του πίνακα knn και τον ταξινομώ με την qsort. Ο knn είναι αρχικοποιημένος με τιμές inf, ώστε οι αποστασεις distance να είναι πάντα μικρότερες.

```
if (dis == 0){      // don't want to calculate the distance of each element with itself (dis = 0)
    continue;
}

if (k<=K-1){
    knn[i][k].distance = dis;
    knn[i][k].element = j;
    qsort(&knn[i][0], K, sizeof(knn_struct), Asc);
    //printf("i=%d, j=%d, k=%d, knn[%d][%d].distance = %lf\n", i, j,k,i,k, knn[i][k].distance);
    k++;          // column counter of knn
}else{
    if (dis < knn[i][K-1].distance){
        knn[i][K-1].distance = dis;
        knn[i][K-1].element = j;
        qsort(&knn[i][0], K, sizeof(knn_struct), Asc);
    }
}
```

Παραλληλοποίηση:

Επιλέγω 4 processes. Για να διασπάσω το αρχείο σε τέσσερα κομμάτια και το κάθε process να διαβάσει μόνο ένα τέταρτο του αρχείου, θεώρησα σαν offset της fseek τον όρο:
(ROWS/P)*COLS*pid*sizeof(double)

```
fseek(fpi, (ROWS/P)*COLS*pid*sizeof(double), SEEK_SET);         // setting fpi according to the prosses's id
```

Όπου ROWS οι γραμμές του αρχείου, COLS οι στήλες, P ο αριθμός των processes, pid το rank του κάθε process και *sizeof(double) διότι η fseek αριθμεί σε bits.

Στη συνέχεια, δημιουργφώ ένα datatype ώστε να κάνω εφικτή τη μεταφορά του stuct μου μέσω των εντολών send/recv.

```
// Struct Derived Data Type
MPI_Datatype knn_type, oldtypes[2];     // required variables for the send/recv
int blockcounts[2];                     // 2 different types included in the struct
MPI_Aint offsets[2], extent;            // MPI_Aint type used to be consistent with syntax of MPI_Type_extent routine

// Setup description of the 1 MPI_DOUBLE field distance
offsets[0] = 0;
oldtypes[0] = MPI_DOUBLE;
blockcounts[0] = 1;

// Setup description of the 1 MPI_INT field element
// Need to first figure offset by getting size of MPI_DOUBLE
MPI_Type_extent(MPI_DOUBLE, &extent);
offsets[1] = 1 * extent;
oldtypes[1] = MPI_INT;
blockcounts[1] = 1;

// Define structured type and commit it
MPI_Type_struct(2, blockcounts, offsets, oldtypes, &knn_type);      // count, blocklens[], offsets[], old_type, &newtype
MPI_Type_commit(&knn_type);                                         // Commits new datatype to the system
```

Ο νέος τύπος δεδομένωω περιλαμβάνει έναν double ποθ ακολουθείται από ένα int.

Blocking:

Η μεταφορά τον δεδομένω πρέπει να γίνει κυκλικά. Ο MASTER = 0 δίνει στον pid = 1 και δέχεται απο τον προηγούμενο (pid = pnumber-1 = P-1).

Γενικά, καθε process δέχεται απο το αμέσως προηγούμενο και στέλνει στο αμέσως επόμενο.

```
start = MPI_Wtime();
if (pid != MASTER){
    MPI_Recv(&knn[i][0], K, knn_type, pid-1, 0, MPI_COMM_WORLD, &myStatus);     // pid receives i row from pid - 1
    printf("Process %d received from process %d\n", pid, pid - 1);
}

MPI_Send(&knn[i][0], K, knn_type, (pid + 1) % pnumber, 0, MPI_COMM_WORLD);      // pid sends to pid + 1 (last process sends

// Now MASTER process can receive from the last process
if (pid == MASTER){
    MPI_Recv(&knn[i][0], K, knn_type, pnumber-1, 0, MPI_COMM_WORLD, &myStatus); // MASTER receives from last process
    printf(" MASTER Process %d received from process %d\n", pid, pnumber - 1);
}
end = MPI_Wtime();
time += end - start;
```

Αφού γίνει το send περιμένω να σιγουρευτώ ότι το μήνυμα έφτασε.

Non-Blocking:

Η διαδικασία είναι η ίδια με τον blocking αλγόριθμο, με μοναδική διαφορά την χρήση των Isen/Irecv. Η Isend θα στείλει το μήνυμα αλλά δεν ξέρει αν έφτασε στον παραλήπτη. Για το λόγο αυτό προσθέτουμε “wait” ΄ώστε να αποτρέψουμε δρομήσεις και σφάλματα κατά την επικοινωνία.

```
start = MPI_Wtime();
if (pid != MASTER){
    MPI_Irecv(&knn[i][0], K, knn_type, pid-1, 0, MPI_COMM_WORLD, &recv_request);
    MPI_Wait(&recv_request, &myStatus);
    printf("Process %d received from process %d\n", pid, pid - 1);
}

MPI_Isend(&knn[i][0], K, knn_type, (pid + 1) % pnumber, 0, MPI_COMM_WORLD, &send_request);
MPI_Wait(&send_request, &myStatus);

// Now MASTER process can receive from the last process
if (pid == MASTER){
    MPI_Irecv(&knn[i][0], K, knn_type, pnumber-1, 0, MPI_COMM_WORLD, &recv_request);
    MPI_Wait(&recv_request, &myStatus);
    printf(" MASTER Process %d received from process %d\n", pid, pnumber - 1);
```

Μετρήσεις από Hellasgrid:

Για τη μέτρηση του χρόνου χρησιμοποίηση τις MPI_Barrier που περιμένει τοπρόγραμμα μέχρι να φτάσουν σε εκείνο το σημείο όλα τα processes. Και την Wtime για την καταγραφή του χρόνου τη στιγμή που καλείται. Έγιναν μετρήσεις και για το συνολικό κώδικα και για το κομμάτι στης επικοινωνίας ξεχωριστά.

Μορφή: i element’s K nearest neighbors are: j + distance

Στην blocking ο χρόνος του αλγορίθμου είναι περίπου 4105 δευτερόλεπτα, ενώ για το data passing μόνο 13 δευτερόλεπτα.

```
59995 element's 5 nearest neighbors are:  57228 + 0.000138    44032 + 0.000139
  57227 + 0.000147    54073 + 0.000150    59999 + 0.000152

59996 element's 5 nearest neighbors are:  55780 + 0.000108    46041 + 0.000108
  54214 + 0.000143    58952 + 0.000144    55746 + 0.000150

59997 element's 5 nearest neighbors are:  55784 + 0.000062    58178 + 0.000072
  57323 + 0.000074    54974 + 0.000100    54525 + 0.000103

59998 element's 5 nearest neighbors are:  57291 + 0.000072    59999 + 0.000100
  56948 + 0.000117    54293 + 0.000119    54177 + 0.000119

59999 element's 5 nearest neighbors are:  57291 + 0.000080    56339 + 0.000084
  59998 + 0.000100    59992 + 0.000108    58041 + 0.000108

Clock time of MASTER's data passing (pid=0) = 13.241910
Clock time of program = 4104.827262

End

========================================================================
```

Στην non-blocking ο συνολικός χρόνος είναι (μετρούμενος από τον MASTER = 0) είναι 4352’’, ενώ για το data passing είναι 741’’.

```
59995 element's 5 nearest neighbors are:  57228 + 0.000138    44032 + 0.000139
  57227 + 0.000147    54073 + 0.000150    59999 + 0.000152

59996 element's 5 nearest neighbors are:  55780 + 0.000108    46041 + 0.000108
  54214 + 0.000143    58952 + 0.000144    55746 + 0.000150

59997 element's 5 nearest neighbors are:  55784 + 0.000062    58178 + 0.000072
  57323 + 0.000074    54974 + 0.000100    54525 + 0.000103

59998 element's 5 nearest neighbors are:  57291 + 0.000072    59999 + 0.000100
  56948 + 0.000117    54293 + 0.000119    54177 + 0.000119

59999 element's 5 nearest neighbors are:  57291 + 0.000080    56339 + 0.000084
  59998 + 0.000100    59992 + 0.000108    58041 + 0.000108

Clock time of MASTER's data passing (pid=0) = 741.250419
Clock time of program = 4352.191525

End

========================================================================
```

Η παραλληλοποίηση μειώνει σε σημαντικό βαθμό τον χρόνο εκτέλεσης σε σχέση με τον σειριακό κώδικα, ειδικά για την επεξεργασία πολλών δεδομένων.

Επίσης, η μεταφοράς δεδομένων send/recv της blocking είναι αρκετά πιο γρήγορη, όμως η non-blocking είναι πιο ασφαλής μέθοδος. Η blocking μπαίνει πιο εύκολα σε deadlock για μεγάλο πλήθος δεδομένων.

Η χρήση τεσσάρων processes είναι σημαντική γιατί αποφεύγουμε την είσοδο σε deadlock ή δρομήσεων, σε αντίθεση με τα πέντε processes.

Τέλως, όπως προανέφερα αντί να αποθηκεύσω το file μέσα σε ένα πίνακα 60000x30, επέλεξα την χρήση δύο fp για το διάβασμα του αρχείου. Θυσιάζοντας χρόνο, κέρδιασα χώρο (δεσμεύω μόνο 2x60).

Blocking:
https://www.dropbox.com/s/ttltp867l1zxw4g/knn_mpi_blocking.c?dl=0

Non-blocking:
https://www.dropbox.com/s/v7t8bym51e03mh8/knn_mpi_non_blocking.c?dl=0

Serial: https://www.dropbox.com/s/2s1by3u7s3ue4lo/knn_Serial.c?dl=0