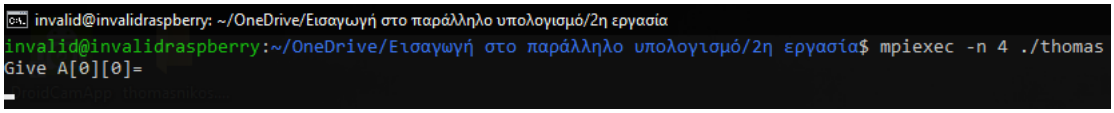
Πανεπιστήμιο Δυτικής Αττικής
Σχολή Μηχανικών
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

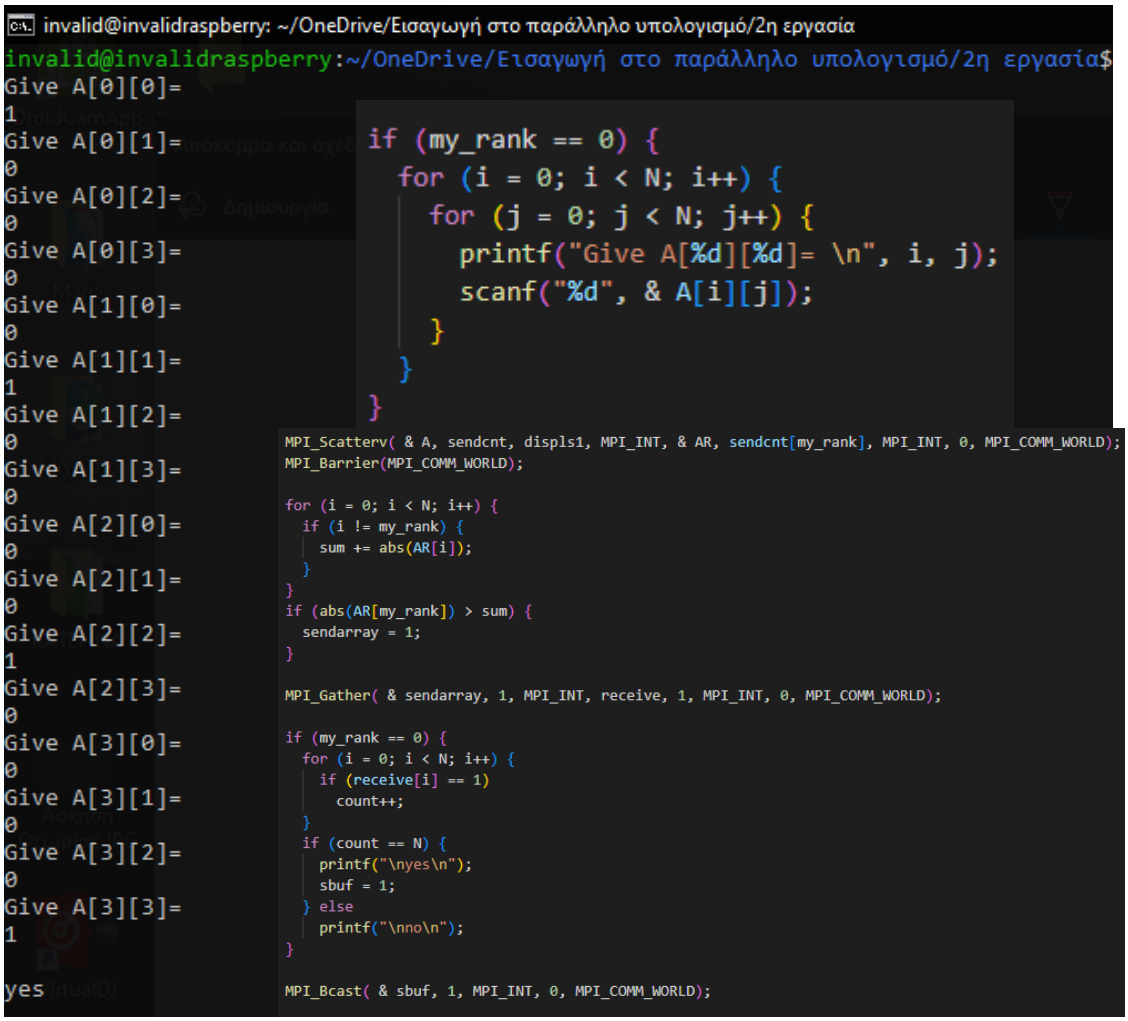
Εργαστήριο Εισαγωγής στον παράλληλο υπολογισμό

ΝΙΚΟΛΑΟΣ ΘΩΜΑΣ
ΑΜ: 21390068
ΤΜΗΜΑ: Ε8

ΑΘΗΝΑ
Τετάρτη, 17 Ιανουαρίου 2024



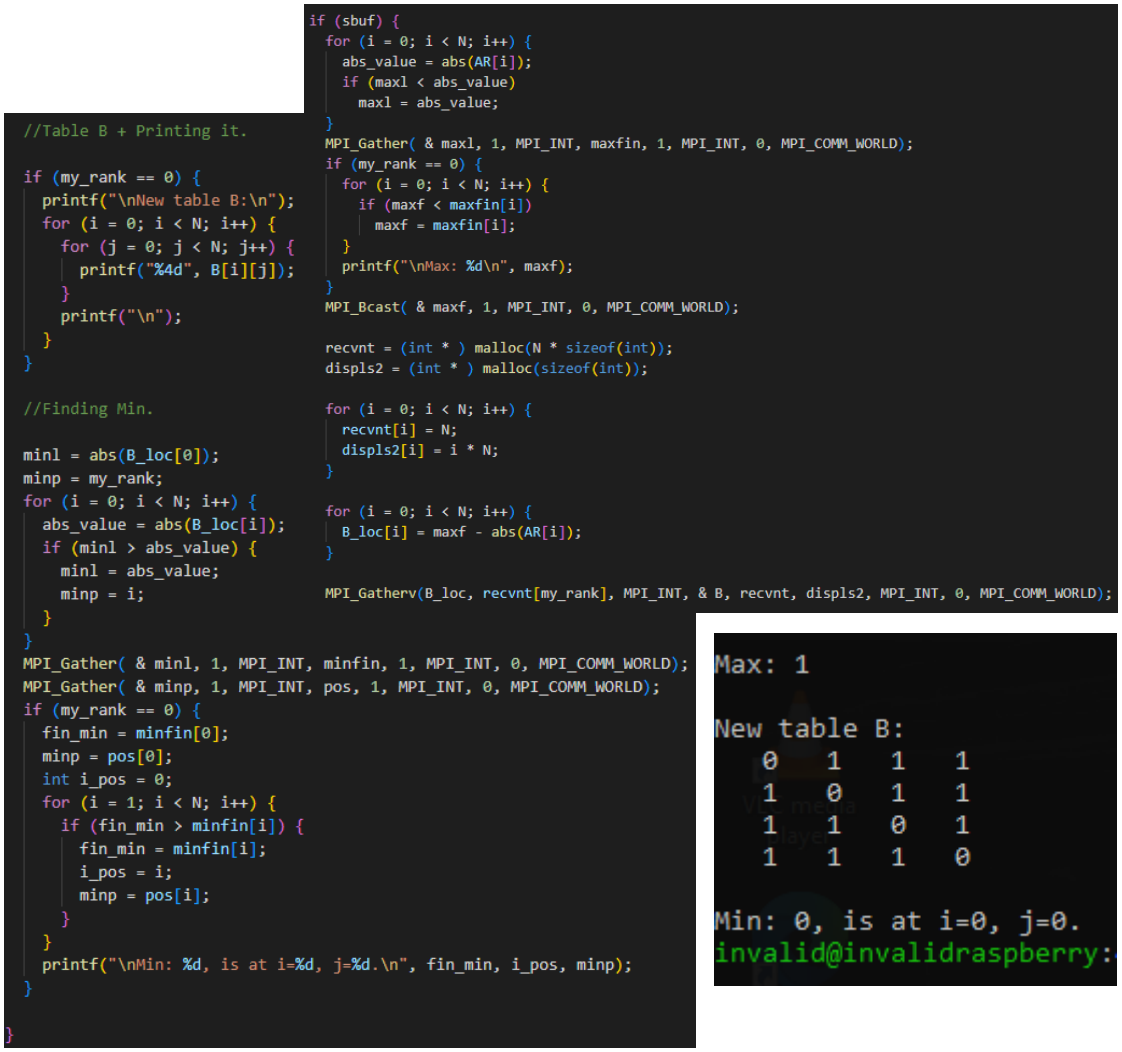
Εικόνα 1: Running the ex.



Εικόνα 2: Ορισμός του πίνακα Α από τον χρήστη

Απάντηση για το αν ο πίνακας είναι αυστηρά διαγώνια δεσπόζων.

Στο παραπάνω στιγμιότυπο οθόνης εμφανίζεται ο κώδικας που χρησιμοποιείται για να δώσει ο χρήστης το πίνακα Α. Αν είναι η διεργασία 0, τότε ξεκινάει ζητώντας από τον χρήστη το κάθε αριθμό του πίνακα με τη σειρά. Διαβάζει όλους τους αριθμούς και τους βάζει σε ένα int array. Στην συνέχεια στέλνει το array σε κάθε διεργασία (Scatterv). Χρησιμοποιούμε το MPI_Barrier για να εξασφαλίσουμε ότι καμία διεργασία δεν θα προχωρήσει προς τα εμπρός πέραν του σημείου αυτού μέχρι όλες οι διεργασίες να έχουν φτάσει σε αυτήν τη συγκεκριμένη στιγμή εκτέλεσης. Με το MPI_Gather κάθε διεργασία παίρνει το αποτέλεσμα της if.



```
if (sbuf) {
  for (i = 0; i < N; i++) {
    abs_value = abs(AR[i]);
    if (maxl < abs_value)
      maxl = abs_value;
  }
  MPI_Gather( & maxl, 1, MPI_INT, maxfin, 1, MPI_INT, 0, MPI_COMM_WORLD);
  if (my_rank == 0) {
    for (i = 0; i < N; i++) {
      if (maxf < maxfin[i])
        maxf = maxfin[i];
    }
    printf("\nMax: %d\n", maxf);
  }
  MPI_Bcast( & maxf, 1, MPI_INT, 0, MPI_COMM_WORLD);

  recvnt = (int * ) malloc(N * sizeof(int));
  displs2 = (int * ) malloc(sizeof(int));

  for (i = 0; i < N; i++) {
    recvnt[i] = N;
    displs2[i] = i * N;
  }

  for (i = 0; i < N; i++) {
    B_loc[i] = maxf - abs(AR[i]);
  }

  MPI_Gatherv(B_loc, recvnt[my_rank], MPI_INT, & B, recvnt, displs2, MPI_INT, 0, MPI_COMM_WORLD);
  //Table B + Printing it.

  if (my_rank == 0) {
    printf("\nNew table B:\n");
    for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++) {
        printf("%4d", B[i][j]);
      }
      printf("\n");
    }
  }

  //Finding Min.

  minl = abs(B_loc[0]);
  minp = my_rank;
  for (i = 0; i < N; i++) {
    abs_value = abs(B_loc[i]);
    if (minl > abs_value) {
      minl = abs_value;
      minp = i;
    }
  }
  MPI_Gather( & minl, 1, MPI_INT, minfin, 1, MPI_INT, 0, MPI_COMM_WORLD);
  MPI_Gather( & minp, 1, MPI_INT, pos, 1, MPI_INT, 0, MPI_COMM_WORLD);
  if (my_rank == 0) {
    fin_min = minfin[0];
    minp = pos[0];
    int i_pos = 0;
    for (i = 1; i < N; i++) {
      if (fin_min > minfin[i]) {
        fin_min = minfin[i];
        i_pos = i;
        minp = pos[i];
      }
    }
    printf("\nMin: %d, is at i=%d, j=%d.\n", fin_min, i_pos, minp);
  }

}
```

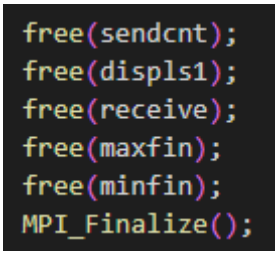
```
Max: 1

New table B:
   0   1   1   1
   1   0   1   1
   1   1   0   1
   1   1   1   0

Min: 0, is at i=0, j=0.
invalid@invalidraspberry:
```

Εικόνα 3: Πίνακας B, Max, Min.

Στο παραπάνω στιγμιότυπο οθόνης εμφανίζεται ο κώδικας που χρησιμοποιείται για την εύρεση του min του max και την δημίουργια του νέου πίνακα B με περιορισμούς $B_{ij} = m - |A_{ij}|$ *για* $i<>j$ *και* $B_{ij} = m$ *για* $i=j$.



```
free(sendcnt);
free(displs1);
free(receive);
free(maxfin);
free(minfin);
MPI_Finalize();
```

Εικόνα 4: Free() και MPI_Finalize().

Απαραίτητη η αποδέσμευση της μνήμης (Free) και η ολοκλήρωση του MPI (MPI_Finalize).