أحمد محمد فتحي عثمان

شرح المشروع

بسم الله

بدأية تم بناء المشروع على جهاز Dell Core i5
بنظام تشغيل Arch Linux
وأتمنى أن لا يكون هناك أي مشكلة في الكود بسبب أي
أختلاف في ال Development Environment

1- بدأت بتعريف متغير world_rank لترتيب ال nodes
حسب ال MPI_COMM_WORLD
2- ومتغير world_size لعدد ال nodes في ال
MPI_COMM_WORLD

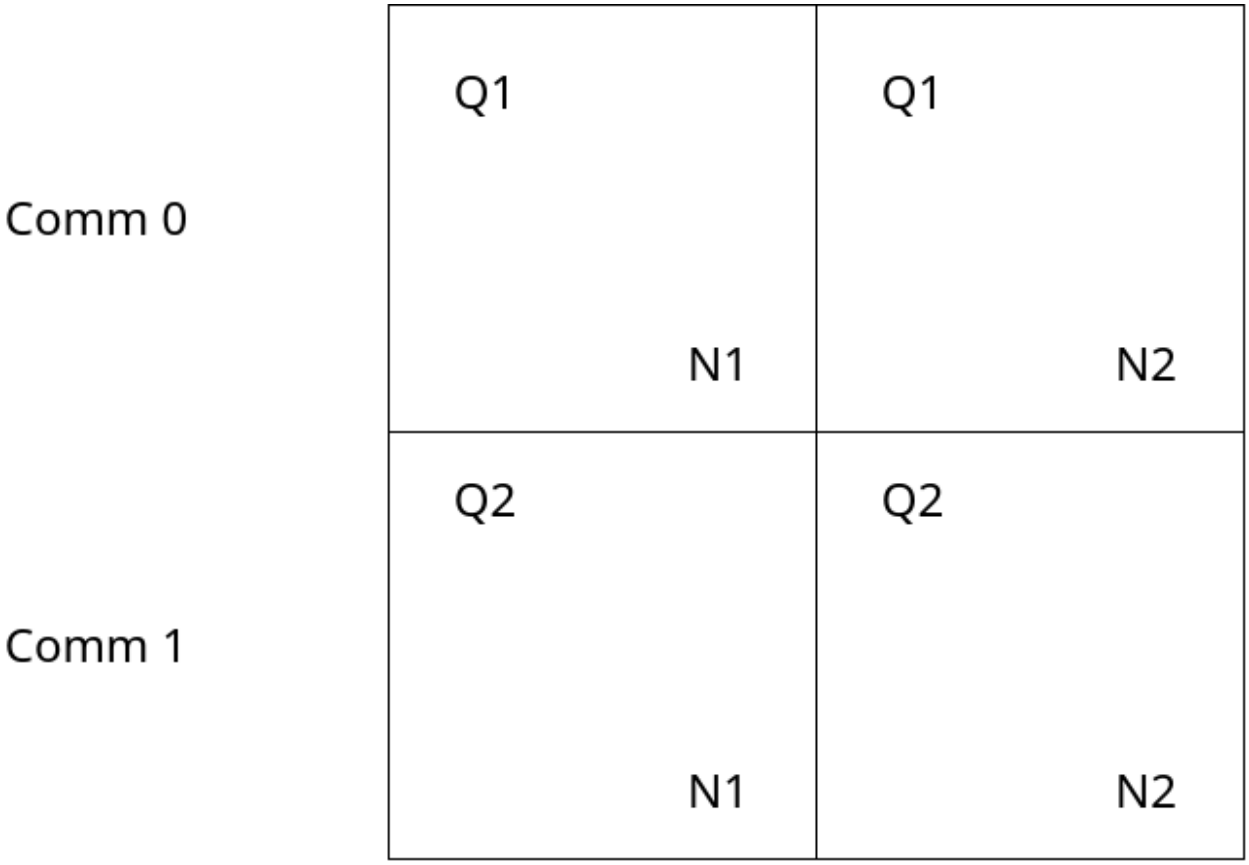
3- المتغير color يستخدم لحساب ال node الحالية تنتمي

إلى أي communicator
4- ثم دمج ال node إلى ال communicator الحالي وهو row_comm

5- المتغير row_rank لترتيب ال nodes حسب ال row_comm
6- ومتغير row_size لعدد ال nodes في ال row_comm

7- نبدأ بقرأة ال database في حالة إذا كنا في ال node 0 بالنسبة لل MPI_COMM_WORLD أي world_rank يساوي 0 ونقرأ ال database بواسطة get_database وترجع database_size

8- وكذالك الأمر بالنسبة لل queries نقرأها في حالة world_rank يساوي 0 ونقرأ ال queries بواسطة get_queries وترجع queries_size



9- بقسم ال database على ال communicators بالطريقة كما في الصورة السابقة حيث أن N1 هو النصف الأول من database و N2 هو النصف الثاني

وكما بالصورة فإن ال database تكون مكررة في كل communicator ولكن منقسمة في كل communicator بحيث أن أول node في ال communicator يكون بها النصف الأول وثاني node يكون بها النصف الثاني

10- على العكس بالنسبة لل queries حيث أن Q1 هو

النصف الأول من queries و Q2 هو النصف الثاني

فكما بالصورة يحتوي ال communicator الأول على النصف الأول من ال queries وال communicator الثاني يحتوي على النصف الثاني من ال queries
ويكون النصف مكرر في كل node من ال communicator بحيث ال node الأولى والثانية تحتوي على نفس النصف

إستخدمت لتقسيم ال database وال queries بين ال communicators وبعضها MPI_Recv و MPI_Send
وللتقسيم بين كل node والأخرى في ال queries فانكشن MPI_Bcast ولتقسيم ال database استخدمت MPI_Scatter و MPI_Bcast

11- وأقسم ال database على ال communicator من خلال
distribute_database_between_communicators

12- وأقسم ال queries على ال communicators من خلال
distribute_queries_between_communicators

13- نبدأ من هنا إجراء الAlgorithm الخاص بنا على ال nodes وتخزين القيم الناتجه لكل query
مع مراعة الترتيب لإختيار أقل 10 words تحتاج عدد معين من ال operations لتصبح نفس الكلمة من ال query عن طريق calculate_levenshtein_distance

14- وLevenshtein Distance Algorithm
من الخوارزميات المشهورة وكثيرة الإستخدام في علوم الحاسب
إستعنت بالله ثم بهذا المقطع لفهم الخوارزمية إضغط هنا
وطريقة تطبيقها لحل المشاكل العمليه
وخاصة أني وجدت مشكلة مثلها على أشهر مواقع ال Problem Solving وهو LeetCode هنا مكان المسألة
إضغط هنا
وهذا هو حلي للمسألة بإستخدام ال Dynamic

Programming إضغط هنا

15- وبدأت في إستخدام ال OpenMP على ال for في كل loop ولم أرد تحديد عدد معين من threads بحيث يستخدم القيمة ال default حسب الجهاز في حالتي يكون ال num_thread(4) / thread_count = 4 لان عدد ال cores بالجهاز هو 4

16- نبدأ بتجميع عشر كلمات لكل query إذا حولتها إلى ال query تحتاج أقل عدد من ال operations من كل ال nodes في ال communicator بإستخدام MPI_Gather مع ترتيبهم

17- ثم نجمع نتائج كل ال queries في ال node التي لها world_rank يساوي صفر بإستخدام MPI_Recv و MPI_Send وإعادة database_size و queries_size الى قيمهم الأساسية
وكل هذا يتم في ال MPI_COMM_WORLD

18- في النهاية نعرض أقل عشر قيم من ال operations تحتاجها بعض الكلمات من ال database لتصبح كلمة معينه حسب كل query

19- ولا ننسى MPI_Comm_free لتحرير المساحات المستخدمة فى كل communicator تم إنشائه

اللهم لك الحمد في اليسر والعسر
اللهم لك الحمد على نعمك التي لا يحصيها غيرك