1. Partie Concurrente (Go) [8 points][8% de votre note finale]

In the Go version of your project, we ask you to concurrently run the DBSCAN algorithm on partitions of the Trip Record data. To create these partitions, you will divide the geographical area into a grid of NxN. The following figure illustrates the case of a partition made of 4x4 cells:

Pour la version Go de votre projet, nous vous demandons d'exécuter, de façon concurrente, l'algorithme DBSCAN sur des partitions des données de courses de taxis. Afin de créer ces partitions, vous devez subdiviser le secteur géographique en une grille de NxN cellules. L'exemple ci-dessous montre une partition composée de 4x4 cellules :

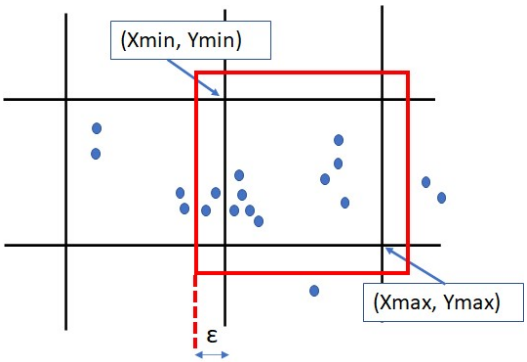
In this case you would then have to run 16 concurrent DBSCAN threads. The first step will therefore be to assign each GPS coordinate to its partition. This is the Map step. But since you will have to combine all your results, you need to slightly expand each of your cell by adding an eps value around it. This way the clusters of adjacent cells will eventually intersect.

For example if you have a grid cell delimitated by the GPS points (x_{min}, y_{min}) *and* (x_{max}, y_{max})*, you will accept all points located inside the expanded cell* $(x_{min} - \varepsilon, y_{min} - \varepsilon)$ *and* $(x_{max} + \varepsilon, y_{max} + \varepsilon)$*. This means that some points will be assigned to more than one partition (refer to figure on the next page).*

Once the clusters identified in each partition using DBSCAN, the last step will be to combine the results to produce a global clustering. This is the Reduce step.

Avec une subdivision telle que montrée à la page précédente, vous auriez à lancer 16 fils concurrents DBSCAN. La première étape sera donc d'attribuer chaque coordonnée GPS à sa partition. Toutefois, puisque ces résultats seront éventuellement combinés, vous devrez élargir chacune des cellules en y ajoutant une valeur eps de tous les côtés. De cette façon, les groupes de cellules adjacentes pourront éventuellement s'intersecter.

Par exemple, si vous considérez une cellule délimitée par les points GPS (x_{min}, y_{min}) et (x_{max}, y_{max}), vous accepterez tous les points à l'intérieur de la cellule élargie $(x_{min}- \varepsilon, y_{min}- \varepsilon)$ et $(x_{max}+ \varepsilon , y_{max} + \varepsilon)$. Conséquemment, certains points se retrouveront à l'intérieur de deux partitions.



Une fois les groupements de points identifiés dans chacune des partitions, la dernière étape consistera à combiner les résultats afin de produire un groupement global.

Algorithme général

The parallel DBSCAN algorithm proceeds then as follows:

1. *Map the data into overlapping partitions (the overlap with other partitions must be at least equals to eps along its border)*
2. *Apply the DBSCAN algorithm over each partition.*
3. *Reduce the results by collecting the clusters from all partitions. Intersecting clusters must be merged.*

This algorithm is based on the MapReduce pattern, widely used in concurrent programming.

*We already explained Step 1. For Step 2, you simply use the regular DBSCAN algorithm but, very important: you must not use the same IDs in different partitions. In the regular algorithm, new clusters are assigned a label that corresponds to the current cluster count, here you have to add an offset that will make sure you will not have duplicate labels. For example, for a new cluster in partition (i,j), the offset could be (10 000 000 * i + 1 000 000 * j), assuming there are less than 1000000 clusters per partition.*

What about Step 3? The principle is simple: we have clusters for each individual partition but since one cluster can cover more than one partition, the intersecting clusters have to be merged. To do so, we just have to consider the points that belong to two adjacent partitions (i.e. the ones located inside the added margin around the cell), if this point is associated with two clusters (one from each grid cell), then these two clusters must be merged. However, in this part of the comprehensive assignment, you do not have to program this step. We will leave to a later exercise…

L'algorithme parallèle du DBSCAN procède comme suit :

1. Distribuer les données à travers les partitions. Ces partitions doivent s'intersecter d'une valeur au moins égale à epsilon.
2. Appliquer l'algorithme DBSCAN sur chaque partition.
3. Réduire les résultats par la collecte de tous les groupements de toutes les partitions; les groupes en intersection doivent être fusionnés.

Cette approche se base sur le modèle MapReduce largement utilisé en programmation concurrente.

L'Étape 1 a déjà été expliqué dans la section précédente. Pour l'Étape 2, il s'agit de simplement utiliser l'algorithme DBSCAN sur chacune des partitions. Mais, il est important de ne jamais utiliser le même identificateur (ID) de groupes à travers les différentes partitions. Dans la version régulière de l'algorithme, chaque nouveau groupement se voit attribuer un ID égal au décompte courant du nombre de groupements. Ici, chaque partition ajoutera une constante à ce nombre afin de ne pas dupliquer les IDs. Par exemple, pour la partition (i,j), cette constante pourrait être (10 000 000 * i + 1 000 000 * j); cette formule sera valide si les partition contiennent moins de 1000000 de groupements.

Qu'en est-il de l'Étape 3? Le principe est simple : nous avons obtenu les groupements pour chacune des partitions mais certains groupes s'étendent sur plus d'une partition. Pour ce faire, il suffit de considérer les points appartenant à plus d'une partition (c'est-à-dire ceux situés sur les régions d'intersection entre partitions). Si un de ces points est associé avec deux groupements différents (un dans chaque cellule) alors ces deux groupements doivent être fusionnés. Toutefois, dans cette partie du projet intégrateur, cette étape n'a pas à être programmée. Laissons cela à un exercice ultérieur…

Le producteur et le consommateur

We ask you to implement a concurrent version of the DBSCAN algorithm that will be based on the producer-consumer pattern.

The producer will be in the main thread and will simply send jobs to a channel; each job being a clustering to be done on one partition of the data. The job instance will include a slice of GPS coordinates and the value of the parameters required to execute the DBSCAN algorithm on this set of points (minPts, eps, offset).

All the jobs are processed by a certain number of consumers, each running in its own thread. When a consumer is done with one job, it then consumes the next job. When there is no more job to consume, then all consumer threads terminate.

Remember that your implementation must include Step 1 and Step 2 but you do not implement Step 3.

Nous vous demandons de réaliser la version concurrente de l'algorithme DBSCAN en vous basant sur le patron de design producteur/consommateur.

Le producteur devra être dans le fil principal et sera simplement chargé d'envoyé les tâches à réaliser à un channel. Ici, une tâche (*job*) correspond à l'application du DBSCAN sur les données d'une partition. L'instance d'une tâche doit donc contenir un slice de coordonnées GPS et les valeur requise pour exécuter l'algorithme sur cet ensemble de points, soient minPts, eps, et la valeur constante pour les IDs.

Toutes ces tâches sont traitées par un certain nombre de consommateurs, chacun s'exécutant dans son propre fil. Lorsqu'une tâche est terminée, la tâche suivante est consommée. Lorsqu'il n'y a plus de tâches à effectuer, alors tous les fils consommateurs se terminent.

Rappelez-vous que seules les Étapes 1. et 2. sont à réaliser. L'étape 3. n'est pas à faire.

Expérimentation

In order to determine the optimal configuration for your concurrent algorithm, we ask you to perform the following experiments and report the execution time for each case:

- *N=2 and 4 consumer threads*
- *N=4 and 4 consumer threads*
- *N=4 and 10 consumer threads*
- *N=10 and 4 consumer threads*
- *N=10 and 10 consumer threads*
- *N=10 and 50 consumer threads*
- *N=20 and 10 consumer threads*
- *N=20 and 50 consumer threads*
- *N=20 and 200 consumer threads*

Also specify the operating system and the specifications of your processor (including the number of cores). You can also add your own experiences with other configuration settings.

For all your experiments, use eps=0.0003 and minpts=5

Afin de déterminer la configuration optimale pour votre algorithme concurrent, nous vous demandons d'effectuer les expériences suivantes :

- N=2 et 4 fils consommateurs
- N=4 et 4 fils consommateurs
- N=4 et 10 fils consommateurs

- N=10 et 4 fils consommateurs
- N=10 et 10 fils consommateurs
- N=10 et 50 fils consommateurs
- N=20 et 10 fils consommateurs
- N=20 et 50 fils consommateurs
- N=20 and 200 fils consommateurs

Pour chacune de ces expériences vous devez spécifier le temps d'exécution ainsi que le système d'exploitation et les spécifications du processeur de la machine utilisée (incluant le nombre de cœurs présents dans ce processeur). Vous pouvez aussi expérimenter d'autres configurations.

Pour toutes vos expériences, utiliser eps=0.0003 et minpts=5.

En plus de votre code source bien commenté, vous devez aussi soumettre un document donnant les résultats des expériences que vous avez réalisées. Remettre aussi les fichiers montrant l'affichage à la console obtenu pour les différentes exécutions de votre programme.