Programació Paral·lela (GED)
Projecte de Laboratori de Pràctiques

Eduardo César
Cristina Peralta
Betzabeth Leon
Sandra Mendez
Computer Architecture and Operating Systems
Universitat Autònoma de Barcelona
Cerdanyola, Spain
{eduardo.cesar, cristina.peralta, betzabeth.leon, sandra.mendez}@uab.cat

***Resum*—El document presenta la pràctica que farem a l'assignatura de Programació Paral·lela durant aquest segon semestre del curs 2023-2024. Amb el doble objectiu de posar en pràctica els coneixements obtinguts en aquesta assignatura i de que adquiriu hàbits de treball autònom, les pràctiques de laboratori es centraran en el desenvolupament d'un únic projecte amb diferents aproximacions.**

Us plantegem un conjunt de fites possibles (implementació de 2 versions paral·leles funcionals del programa que us proporcionem) i uns terminis (lliuraments) en els que haureu d'anar lliurant parts de la feina feta. En aquest document introduïm un cas d'ús de l'algoritme k-means, en particular, la clusterització de píxels per a reduir el nombre de colors diferents d'una imatge.

***Index Terms*—High Performance Computing, aplicacions paral·leles, OpenMP, MPI CUDA, machine learning, supervised clustering.**

I. Introducció

Amb el doble objectiu de posar en practica els coneixements obtinguts en aquesta assignatura i de que adquiriu hàbits de treball autònom, les pràctiques de laboratori es centraran en el desenvolupament d'un projecte obert per la paral·lelització d'un programa real. En aquest projecte us plantegem un conjunt de fites possibles (implementació de versions paral·leles funcionals del programa que us proporcionem) i uns terminis (lliuraments) en els que haureu d'anar lliurant parts de la feina feta.

La planificació necessària per assolir aquestes fites i, fins i tot, decidir quines fites voleu assolir, es responsabilitat vostra.

A la secció II d'aquest document us fem una introducció a l'algoritme de machine learning que farem servir, destacant les característiques més importants del codi original que us proporcionem i que haureu de paral·lelitzar. A la Secció III descrivim les fites del projecte, és a dir, les paral·lelitzacions que us demanem i us donem algunes orientacions i recomanacions generals. Finalment, a la Secció IV s'indiquen els criteris d'avaluació i en quina forma i termini fer els lliuraments.

II. Descripció del problema

El codi que us facilitem es correspon a la implementació en llenguatge C de l'algoritme k-means adaptat per a imatges. L'algoritme k-means es una possible implementació dels algoritmes de clusterització que pertany a la categoria d'algoritmes de machine learning no supervisats. Per tant, les dades que fem servir com a input, s'agruparan segons certs patrons que definirem com a programadors. Un exemple clàssic i força emprat en data science és la clusterització dels clients d'una empresa amb la finalitat de dissenyar diferents accions de màrqueting per a cada grup.

A. K-means

L'algoritme K-means es iteratiu i està dissenyat per particionar un conjunt de dades en un nombre d'agrupacions definides per l'usuari (programador). Aquest nombre d'agrupacions es defineix com a K. Per a cada agrupació (clúster), definim com a centroide el punt representatiu del clúster. Aquest punt es va recalculant a cada iteració amb la mitjana de les dades assignades al clúster en aquesta iteració (mean computation). D'aquí obtenim la expressió K-means.

El primer pas per a realitzar el K-means, consisteix en definir el nombre d'agrupacions o clústers K. Posteriorment, s'han d'inicialitzar els centroides de cada clúster. Es poden fer inicialitzacions aleatòries, distribucions uniformes o inicialitzacions amb valors fixats. En funció de la inicialització es poden obtenir uns resultats o altres ja que es un algoritme sensible a la inicialització, és a dir, no convergeix sempre al mateix estat.

A continuació, per a cada dada del dataset, es computa una funció de proximitat definida per l'usuari, i s'estableix una relació de pertinença a un grup segons un criteri de valor mínim. En altres paraules, a cada dada se li assigna el clúster més proper segons una funció que determina el càlcul de distàncies (Figura 1).

Quan totes les dades han estat assignades, cada agrupació (clúster) computa el nou centroide amb el càlcul de la mitjana de totes les dades assignades a aquest clúster en aquesta iteració.

Aquest procediment d'assignació i actualització de centroides es repeteix fins que cap centroide varia el seu valor en una iteració. Quan cap centroide canvia el seu valor, vol dir que

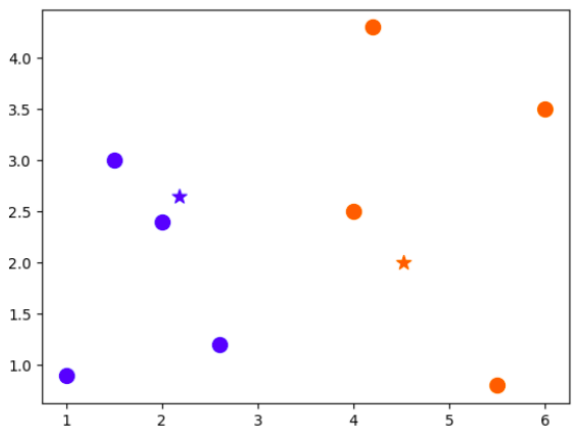

Figura 1. Assignació de punts als clústers segon centroides.

haurem obtingut els valors representatius de cada grup i una assignació única de grup per a cada dada del dataset (Figure 2).

B. Descripció del codi proporcionat

La funcionalitat del codi que us proporcionem consisteix en classificar els píxels d'una imatge que passem com a input, en un nombre definit de clústers K. Una vegada classificats tots els pixels, el programa estableix el color de cada píxel segons el color del centroide del clúster al que pertany. Per tant, des d'un punt de vista d'usuari, estem aplicant un filtre per reduir el nombre de colors diferents d'una imatge, conservant els colors més representatius de la mateixa.

El codi requereix d'una imatge d'extensió bit map color (bmp) i que tingui una profunditat de color de 24 bits. Aquests 24 bits guarden la informació dels 3 valors red, green i blue (RGB), 1 byte per cadascun d'aquests colors (i per tant amb valors de 0 a 255). A partir d'aquesta imatge es generen dos

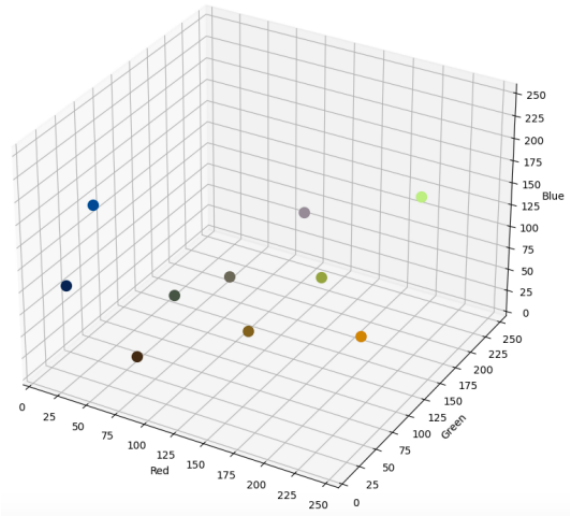

Figura 2. Centroides finals calculats per l'algoritme K-means.

outputs, un fitxer amb els valors dels centroides trobats i la imatge amb la nova assignació de colors.

C. Quantificació del problema

La imatge que us proporcionem per a l'execució del problema[1] és de 3840×2160 píxels; com que cada píxel ocupa 24 bits, tenim una imatge d'uns 24 MB. Els centroides ocupen 19 Bytes cadascun i no en tindrem més de 255, per tant parlem de l'ordre de pocs KB.

Heu de tenir en compte que algunes de les estratègies que implementareu, poden ser més o menys eficients segons la mida de la imatge i el nombre de centroides.

III. Paral·lelització

L'objectiu principal del projecte que us plantegem és el de paral·lelitzar la implementació proporcionada del algoritme fent servir diferents aproximacions: memòria compartida i us d'acceleradors (per pas de missatges farem servir un problema diferent). D'aquesta manera experimentareu quines diferències existeixen entre les versions implementades i obtindreu un coneixement per a decidir en quins contextos implementaríeu solucions en el món empresarial i/o científic.

El segon objectiu consisteix en realitzar una anàlisi del rendiment de l'aplicació. Es farà una anàlisi de comportament de les versions implementades, s'executaran utilitzant diferents configuracions, es prendran mesures corresponents i s'identificarà un o més problemes de rendiment (indicant la seva importància potencial). Farem servir les eines d'anàlisi disponibles (`likwid`, `perf`, etc.).

Es important destacar que no heu de confondre el que és la optimització del codi en sèrie per a un únic processador, amb la preparació del codi per a ser executat en plataformes d'alt rendiment. Un bon codi sèrie pot ser molt complexe i a més a més, pot dificultar la paral·lelització del mateix.

A. Memòria compartida

Farem servir OpenMP per fer aquesta implementació.

En aquest cas, cal decidir en primer lloc si farem una parallelització basada en tasques o en bucles.

En cas de que es faci basada en tasques, caldrà definir que és una tasca en aquest programa i possiblement caldrà modificar de forma considerable el codi.

En cas de que es faci basada en bucles, caldrà identificar quins bucles poden ser paral·lelitzats, quines dependències de dades existeixen i buscar les solucions que introdueixin la menor sobrecàrrega possible.

B. Acceleradors

Farem servir OPENACC (i opcionalment CUDA) per fer aquesta implementació.

En aquest cas, el més important és identificar les seccions de codi que es poden beneficiar d'executar-se a l'accelerador i que, per tant, justifiquen descarregar-les a l'acceleradora o escriure un kernel en CUDA. A més, cal identificar clarament

[1] https://s2.best-wallpaper.net/wallpaper/3840x2160/1807/League-of-Legends-game-characters_3840x2160.jpg

l'operació que es realitza sobre les dades amb l'objectiu d'utilitzar algorismes coneguts (i possiblement ben optimitzats) per les acceleradores.

C. Recomanacions de desenvolupament

Us presentem algunes recomanacions de com desenvolupar la pràctica:

- Teniu a la vostra disposició documents sobre el funcionament dels dos clusters del departament disponibles (cluster del laboratori i cluster Wilma) i l'enviament de treballs al gestor de recursos. Feu-los servir! I si no en teniu prou, aneu als manuals originals.
- Algunes proves poden necessitar molt de temps i a més, en el cas de l'anàlisi de rendiment, poden generar quantitats molt grans d'informació. En conseqüència, pot ser una bona idea generar versions reduïdes del vostre programa (fent només unes poques iteracions, per exemple).
- Organitzar de forma adient el projecte al vostre compte d'usuari i fer servir eines de control de versions (existeixen unes quantes amb opcions gratuïtes):
  - **GitHub** https://github.com/
  - **GitLab** https://about.gitlab.com/

  Així evitarem ensurts per pèrdues d'informació, decisions catastròfiques, etc.
- Fer servir el make, es a dir, escriure Makefiles que permetin generar diferents versions del codi (ex. complerta, reduïda) i netejar el directori de treball d'arxius executables i amb codi objecte, de forma àgil.
- Comenteu el vostre codi amb comentaris útils (sobre tot per vosaltres mateixos). Per tal que un comentari sigui útil, és molt important que vagi acompanyat per la data i l'autor.
- Les proves funcionals dels vostres programes OpenMP, MPI i CUDA es faran de forma remota. Recordeu que cal utilitzar les cues d'execució disponibles, es a dir, la cua `test.q, aolin.q i cuda.q` del laboratori o la cua `nodo.q` del cluster Wilma.
- Les proves per l'anàlisi de rendiment dels vostres programes OpenMP i MPI poden fer-se tant en el cluster Wilma (cua `nodo.q`), com en el cluster del laboratori d'arquitectura (cua `aolin.q`). Pel que fa la pràctica amb CUDA haureu de fer servir la cua `cuda.q` del laboratori d'arquitectura.

IV. Lliurament i Avaluació

Per a cada aproximació de paral·lelització disposeu de 2 sessions de laboratori. A la última sessió de cada part haureu d'explicar oralment les implementacions realitzades. Al finalitzar la última sessió disposeu d'un curt termini per entregar un document explicatiu de la pràctica. Això vol dir que a la última sessió ja hauríeu de tenir tota la documentació i només contrastar amb els/les professors/es petits detalls o observacions que tingueu en les vostres últimes execucions.

La nota total de pràctiques representa un 35% de la nota final de la assignatura. Sent requisit indispensable tenir una nota de 5 o superior en la nota final de pràctiques per aprovar l'assignatura. L'aportació de cada versió es mostra a la taula I.

Taula I
Pes de cada lliurament a la nota final de l'assignatura.

OpenMP	**12%**
MPI	**12%**
OPENACC (CUDA)	**11%**

Durant les pràctiques haureu de realitzar 3 lliuraments corresponents cadascuna amb una versió paral·lela: OpenMP, OPENACC (CUDA)[2] i MPI[3]. Un lliurament consta de:

- Una implementació paral·lela funcional del codi proposat que serà mostrada al/la professor/a de pràctiques abans de cada lliurament (OpenMP, OPENACC (CUDA)).
- Una memòria en format pdf on es descriu:
  - Explicació de forma detallada de com s'ha de compilar i executar el vostre codi.
  - L'estratègia de paral·lelització.
  - Els canvis realitzats al codi.
  - Mètriques de rendiment: l'acceleració (speed up) i eficiència obtinguts respecte a l'original fent servir diferents nombres de recursos.
  - L'anàlisi de rendiment realitzat.
  - El/els problema/es trobat/s, la seva importància i possibles solucions.

El lliurament de l'informe en pdf i el codi font (`*.h, *.c, *.cu`) es farà al Campus Virtual de l'assignatura empaquetat en un arxiu `.zip` o `.tar` fins la data indicada al CV.

Per a calcular la nota dels lliuraments es farà servir el següents criteris:

- [1,5 punts] Explicació de l'estratègia de paral·lelització de forma acurada i argumentada.
- [1 punts] L'estratègia de paral·letització emprada és adient per al codi proposat.
- [2 punts] Les adaptacions realitzades al codi i el seu 'correcte' funcionament.
- [0,5 punts] La documentació i estil del codi.
- [1,5 punts] Obtenció de mètriques amb diferents eines, mides de problema i nombre de recursos.
- [1,5 punts] Anàlisis de rendiment per a les execucions realitzades.
- [0,5 punts] Crítica dels problemes sorgits i les solucions trobades en el desenvolupament de la pràctica.
- [0,5 punts] Avaluació continuada d'assistència i participació a classe [NOTA INDIVIDUAL].
- [1 punts] Respostes donades oralment a la última sessió de cada versió [NOTA INDIVIDUAL].

[2]Aquells que a més de la versió OPENACC feu una prova funcional amb CUDA tindreu nota extra en les pràctiques.

[3]L'enunciat per la pràctica MPI es publicarà més endavant.

Us recordem que per a cada entrega heu d'especificar les comandes de compilació del vostre codi i el compilador que heu fet servir.

V. Conclusions

Ja us hem presentat el projecte que fareu durant aquest semestre. Qualsevol dubte o pregunta que teniu, si us plau, contacteu amb nosaltres Eduardo.Cesar@uab.cat, Cristina.Peralta@uab.cat, Betzabeth.Leon@uab.cat i Sandra.Mendez@uab.cat. No dubteu en demanar tutories!