Résumé

Cette thèse vise pour son essentiel la conception et la mise en œuvre d'algorithmes parallèles temps réels de clustering, réputésd'être très chronophages et énergivores dans la détection des objets abandonnés dans le domaine de la vidéosurveillance. Les GPUs sont très prometteurs pour l'accélération des algorithmes à forte intensité de calcul. Le langage Open Computing Language (OpenCL) est une API (Application Programming Interface) multiplateforme standardisé conçu pour prendre en charge le développement d'applications parallèles sur des systèmes informatiques hétérogènes. OpenCL permet aux développeurs d'utiliser le potentiel des GPUs sans avoir besoin d'un longtemps de développement et de connaissances matérielles approfondies. La plateforme OpenCl est utilisée dans cette thèse pour accélérer les algorithmes à forte intensité de calcul dans le domaine de l'apprentissage automatique et du calcul scientifique. Les algorithmes étudiés sont le clustering k-means, et K-means++. Les noyaux de clustering K-means et K-means++ conçus pour les GPUs ont largement surpassé les implémentations optimisées du CPU.

Les implémentations parallèles des algorithmes du clustering K-means et K-means++, en utilisons la plateforme OpenCL synthétisées pour GPUs, ont obtenu de bons résultats par rapport aux CPU. L'algorithme K-means parallèle en utilisant la plateforme OpenCL a été implémenté dans le système de détection automatique des objets abandonnés. Les résultats montrent qu'il y a un gain de temps qui surpassant de manière significative l'algorithme K-means séquentiel, en gardant la même précision pour la détection des objets abandonnés. Notre système a été conçu pour traiter des vidéos en temps réel pour la détection des objets abandonnés.

Abstract

This thesis focuses on the design and implementation of real-time parallel clustering algorithms known to bevery time-consuming and energy-consuming in the detection of abandoned objects in visual surveillance. GPUs are verypromising for accelerating computationally intensive algorithms. The Open Computing Language (OpenCL) is a standardized cross-platform API designed to support the development of parallel applications on heterogeneous computing systems, OpenCL allows developers to utilize the potential of GPUs without the need for extensive development and hardware knowledge. The OpenCL platform is used in this thesis to accelerate computationally intensive algorithms in the area of machine learning and scientific computing. The studied algorithms are K-means and K-means++ clustering. The K-means and K-means++ clustering kernels designed for GPUs have largelyout performed the CPU-optimized implementations.

Parallel implementations of the K-means and K-mean++ clustering algorithms using the Open Computing Language (OpenCL) platform designed for GPUs have achieved good results compared to CPUs.

The parallel K-means algorithm using the OpenCLplatform was implemented in the system to detect abandoned objects in real-time. The results showed that there was a time gain that significantly outperformed the sequential K-means algorithm while maintaining the same accuracy for abandoned object detection. Our system was designed to process real-time video for abandoned object detection.

الملخص

تركز هذه الأطروحة على تصميم وتنفيذ خوارزميات التجميع (clustering)المتوازية في الوقت الحقيقي(real-time) المعروفة التي تستغرق وقتا طويلا وتستهلك الطاقة في الكشف عن الأشياء المتخلي عنها في مجال المراقبة بالفيديو. وحدات معالجة لرسوماتGPU واعدة جدا لتسريع الخوارزميات المكثفة حسابيا. لغة الحوسبة المفتوحة (OpenCL) هي واجهة برمجة تطبيقات موحدة عبر المنصات مصممة لدعم تطوير تطبيقات متوازية على أنظمة الحوسبة غير المتجانسة ،تسمحOpenCL للمطورين باستخدام إمكانات وحدات معالجة الرسومات دون الحاجة إلى تطوير واسع ومعرفة الأجهزة .يتم استخدام منصةOpenCL في هذه الأطروحة لتسريع الخوارزميات المكثفة حسابيا في مجال التعلم الآلي و الحوسبة العلمية. الخوارزميات المدروسة هيk-meanclusteringوclustering ++k-mean . وقد تفوقتkernelsk-meansclustering و++k-meansالمصممةلGPUsإلى حد كبيرعلى التطبيقات المصممة ل CPU.

الخوارزميات المتوازية منK-meansوclustering ++k-means باستخدام منصة لغة الحوسبة المفتوحة (OpenCL) مصممةلGPUsأداء جيدا بالمقارنة مع وحدات المعالجة مصممةل CPU. تم تنفيذ خوارزميةk-means المتوازية باستعمال النظام الأساسيOpenCLفي نظام الكشف التلقائي عن الأشياء المتخلي عنها في مجال المراقبة بالفيديو. وأظهرتا لنتائج أن هناك تحسين كبير في الوقت الذي تجاوز إلى حد كبير خوارزميةk-meansالتقليدية ،مع الحفاظ على نفس الدقة للكشف عنا لأشياء المتخلي عنها. وقد تم تصميم نظام نالمعالجة أشرطة لفيديو في الوقت الحقيقي(real-time) للكشف عن الأشياء المتخلي عنها.

Déclaration des publications

Cette thèse comprend trois articles originaux qui sont publiés dans des revues à comité de lecture, comme suit:

Chapitre de thèse	Titre de publication / citation complète	Statut de publication *
Chapitre3	-S. Daoudi, C. M. A. Zouaoui, M. C. El-Mezouar and N. Taleb, "A Comparative study of parallel CPU/GPU implementations of the K-Means Algorithm," *2019 International Conference on Advanced Electrical Engineering (ICAEE)*, Algiers, Algeria, 2019, pp. 1-5. Published in: IEEE. DOI: 10.1109/ICAEE47123.2019.9014783	Publié
	-S. Daoudi, C. M. A. Zouaoui, M. C. El-Mezouar and N. Taleb, .Accelerating K-Means algorithm with Parallel Implementations using CPU and GPU. International Journal of Electronic and Electrical Engineering Systems .Volume 3 , N°1 – March 2020.https://ijeees.com/index.php/issue-n-1-2020/	Publié
Chapitre4	- S. Daoudi, C. M. A. Zouaoui, M. C. El-Mezouar and N. Taleb. (2021). Parallelization of the K-means++ clustering algorithm. Ingénierie des Systèmes d'Information, Vol. 26, No. 1, pp. 59-66. https://doi.org/10.18280/isi.26010	Publié

Les résultats attendus de cette thèse a permis la production des articles de conferances suivants:

1. «Parallelisation de l'algorithme du K-Means avec OpenCl » Deuxièmes Journées Doctorales de Génie Electrique 4-5 Décembre 2018 Université DjillaliLiabes-Faculté de Génie Electrique.Sidi Bel Abbes, Algerie.
2. « Parallel implementation of the k-means clustering algorithm based on CPU/GPU »"2nd Conference on Informatics and Applied Mathematics" – "IAM Conference 2019" - Université Guelma, Algerie.

3. « A Comparative study of parallel CPU/GPU implementations of the K-Means Algorithm »"the International Conference on Advanced Electrical Engineering 2019 (ICAEE 2019) " Alger, Algerie.

4. «Parallelisation de l'algorithme du K-means ++». Troisièmes Journées Doctorales de Génie Electrique, Faculté de Génie Electrique, UDL Sidi Bel Abbes, 17-18 Décembre 2019

5. «Parallelization of the K-Means++ Clustering Algorithm on GPU» ACM - 8th International Conference on Software Engineering and New Technologies. Hammamet, Tunisia - 28 – 30 December 2019.

Dédicaces

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tout simplement que je dédie ce travail à :

La mémoire de mon très cher grand père qui était et sera toujours ma fierté

Ma très chère mère et cher père

Ma très chère sœur

Mes chers frères

Mes chers (es) amis (es) pour tous les bons moments qu'on a passé ensemble

Tous ceux qui ont contribué de près ou de loin pour que ce travail soit terminé.

Remerciements

Le travail, présenté dans cette thèse de doctorat, a été réalisé au laboratoire de Réseaux de Communications, Architectures et Multimédia Appliquée RCAM de l'université Djillali Liabes de Sidi Bel Abbes.

Je tiens à remercier en tout premier lieu notre Dieu ALLAH le Tout Puissant, Créateur de la terre et de l'univers, de m'avoir donné le courage, la volonté et la patience pour mener à terme ce présent travail.

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à Monsieur BOUNOUA Abdennacer, Professeur à l'Université Djillali Liabes de Sidi Bel Abbes, qui m'a fait l'honneur d'accepter la présidence du jury de cette thèse.

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à Monsieur BELLOULATA Kamel Professeur à l'Université DjillaliLiabes de Sidi Bel Abbes ,quia manifesté l'amabilité d'examiner ce travail.

J'exprime ma reconnaissance à Monsieur BENSLIMANE Sidi Mohamed Professeur à l'école superieur en informatique de Sidi Bel Abbes et directeurde l'ESI-SBA, qui m'a fait un grand honneur en acceptant de juger le travail de cette thèse.

Je voudrais tout particulièrement exprimer mes sincères reconnaissances et ma profonde gratitude à Monsieur Nasreddine TALEB, Professeur à l'Université Djillali Liabes de Sidi Bel Abbes et directeur du laboratoire RCAM, de m'avoir accueilli dans ce laboratoire de recherche. Aussi, je tiens à lui présenter mes remerciements les plus élogieux de m'avoir honoré par sa présence au jury autant qu'invité.

Je tiens à remercier Monsieur CHIKR EL MEZOUAR Miloud, Professeur à l'université Djillali Liabes de Sidi Bel Abbes, pour avoir accepté de diriger cette thèse. Il m'a fait bénéficier de son expérience et de sa rigueur scientifique. Il m'a apporté une aide considérable tout au long de cette étude, ses conseils judicieux et ses encouragements ont été bénéfiques pour mener à bien le travail et la rédaction de cette thèse.

Je voudrais tout particulièrement exprimer mes sincères reconnaissances et ma profonde gratitude à mon Co-encadreur Monsieur ZOUAOUI ChakibMustapha Anouar, Maitre de conférences à l'université Djillali Liabes de Sidi Bel Abbes, pour ses conseils scientifiques, sa disponibilité et son encouragement, ainsi que pour son aide précieuse pour mener ce travail à son terme.Enfin, je tiens à remercier, tous mes amis (es) pour leurs encouragements et leurs soutiens durant nos années d'étude.

Table des matières

Liste des Tableaux

Liste des Figures :

Liste des acronymes:

AO	Abandoned object.
API	Application Programming Interface
ASIC	Application Specific Integrated Circuits
BB	Bounding box
C++ AMP	C++ Accelerated Massive Parallelism
CCTV	Closed-Circuit Television
CPU	Central Processing Unit
CUDA	ComputeUnifiedDevice Architecture
DESL	Domain Embedded Specific Language
DSP	Digital Signal Processor
FPGA	Field Programmable GateArray
GPU	Graphics Processing Unit
Intel TBB	Intel Threading Building Blocks
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
NMS	Non Maximum Suppression
PThreads	Posix threads
OpenACC	Open Accelerators
OpenCL	Open ComputingLanguage
OpenGL	Open Graphics Library
OpenMP	Open Multi-Processing
OpenMPI	Open Message Passing Interface
SIMD:	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SPIR-V	Standard Portable IntermediateRepresentation
SPMD	Single Program Multiple Data.
SSE	Streaming SIMD Extension

Introduction génerale

Contexte

Depuis l'invention du premier circuit intégré en silicium, les performances et les capacités des microprocesseurs ont augmenté de façon exponentielle. En 1965, Golden Moore avait prédit [1] que le nombre de transistors dans un seul circuit intégré doublerait tous les dix-huit mois. Au cours des cinquante dernières années, les tendances en matière de développement des semi-conducteurs ont prouvé qu'il avait raison. En conséquence, des microprocesseurs dotés d'une puissance de calcul exceptionnelle sont devenus de plus en plus rentables.

Cependant, ces dernières années, il est devenu de plus en plus difficile de réduire la taille des transistors [2]. Parallèlement, la demande des microprocesseurs haute performance augmente, en raison des applications émergentes dans divers domaines tels que le calcul mobile, Machine learning, le data mining et l'infographie. À ce jour, le simple ajout de dispositifs de calcul et de mémoire dans un processeur n'est peut-être plus le meilleur moyen d'augmenter les performances. Des solutions alternatives plus intelligentes seront donc nécessaires. La récente génération de réseaux de portes programmables (FPGA) avec des blocs DSP en virgule flottante intégrés et les plates-formes de calcul traditionnelles basées sur les CPU et les GPU permet d'accélérer les problèmes de calcul intensifs.

Machine learning est l'un des domaines de l'informatique qui se développe le plus rapidement aujourd'hui, et dont les applications couvrent toutes ces facettes de notre vie quotidienne. Machine learning est déjà appliqué dans des domaines tels que les moteurs de recherche, le data mining, la vision par ordinateur, le traitement du langage naturel, la robotique, science médicale et commerce, avec de nouvelles applications qui sont découvertes chaque jour. Le domaine du Big Data est très prisé des chercheurs et des compagnies qui développent de nouveaux outils pour analyser des données énormes qu'ils ne pouvaient pas analyser auparavant. Le clustering représente souvent la première étape de l'extraction d'informations car la plupart des algorithmes de clustering fonctionnent de manière non supervisée. Le Clustering est une méthode importante de data mining qui aide les analyseurs à comprendre le regroupement des attributs de données. Le clustering est largement utilisé dans de nombreux domaines tels que data mining, machine learning, la reconnaissance de formes, segmentation d'images, l'intelligence artificielle, l'analyse des données d'expression génétique et labioinformatique. L'un de ces algorithmes du clustering les plus simples et les plus facile est

le K-means (algorithme de Lloyd) et son extension k-means++. Cependant, la plupart des algorithmes clusterings sont intensifs en calcul. Dans de nombreuses applications du clustering, l'exécution des programmes parallélisés sur le GPU pourrait donner une grande vitesse par rapport aux programmes séquentiels ou multi-threads sur le CPU. Lorsque nous étudions leparallélisme du k-means et k-means++ avec le GPU, nous devons explorer les API spécifiques qui permettent de les utiliser. Par exemple, avec une plate-forme de calcul parallèle, les programmeurs du CUDA [3] (ComputeUnifiedDevice Architecture) peuvent accélérer les applications en exploitant seulement la puissance des GPU Nvidia.En effet, le langage Open ComputingLanguage (OpenCL) est une API multiplateforme standardisée conçue pour prendre en charge le développement d'applications parallèles sur des systèmes informatiques hétérogènes, telles que des processeurs multicœurs, des GPU, des DSP, des FPGA et des ASIC [4, 5, 6, 7, 8]. D'autre part, OpenCL a un modèle de gestion des périphériques plus complexe qui promet la portabilité des codes entre différentes architectures matérielles. Dans ce cadre, l'optimisation reste la problématique majeure lorsqu'un même code doit être déployé sur différentes plateformes. Le programme OpenCL se limite en termes de performance de portabilité [9][10]. Ces dernières années, de nombreuses recherches ont été menées sur la portabilité des algorithmes des clusterings sur des plateformes hétérogènes et parallèles.

Cette thèse vise pour son essentiel la conception et la mise en œuvre des algorithmes parallèles temps réels de clustering réputésd'être très chronophage et énergivore dans la détection des objets abandonnés .Nous sommes intéressés d'introduire le parallélisme de calcul du K-means et k-means++. Pour ce faire, nous utiliserons la plate-forme OpenCL et un nouveau logiciel de parallélisation des GPU pour optimiser le clustering K-means et k-means++. Nous pourrions utiliser ces algorithmes parallèles du clustering pour détecter les objets abandonnées.

Contributions

L'objectif de cette recherche est d'accélérer les applications à forte intensité de calcul telles que les algorithmes de clustering en utilisant OpenCL sur GPU et puis l'appliquer dans la détection des objets abandonnés. Les résultats de cette recherche en termes du débit et du temps de calcul total sont comparés aux plateformes de calcul multicœurs traditionnelles telles que le CPU. Les avantages et les inconvénients de l'OpenCL ainsi que des plates-formes CPU et GPU sont également évalués au cours de cette recherche. Les objectifs de la recherche ont été atteints en sept phases :

1. Les principes fondamentaux de la programmation parallèle pour la plate-forme OpenCL ont été étudiés.
2. Une étude des algorithmes parallélisables à forte intensité de calcul a été menée et des algorithmes appropriés pour l'implémentation utilisant OpenMP et Pthread sur CPU ont été sélectionnés.
3. Une étude des algorithmes parallélisables à forte intensité de calcul a été menée et des algorithmes appropriés pour l'implémentation utilisant OpenCL sur GPU ont été sélectionnés.
4. Ces algorithmes ont été implémentés sur GPU et leur exactitude a été vérifiée avec les implémentations CPU.
5. Des améliorations ont été apportées aux implémentations GPU de base afin d'obtenir les meilleures performances possibles.
6. Les meilleures versions des algorithmes implémentés sur GPU ont été testées avec les implémentations CPU disponibles afin de comparer les performances et l'efficacité.
7. La détection d'objets abandonnés est une tâche cruciale pour assurer la sécurité publique. Pour cela nous proposons un nouveau système qui permet la détection de bagages abandonnés dans des lieux publics. L'objectif de ce système étant de délimiter avec précision les frontières des objets pour les classifier comme étant des objets abandonnés, K-means jouent un rôle crucial pour leur capacité à séparer clairement les différentes zones. La nouveauté de notre méthode est l'utilisation de l'algorithme parallèle du clustering k-means en utilisant la plateforme OpenCL .Le traitement doit être effectué par le GPU et le CPU.

Structure de la thèse

Cette thèse est structurée de la manière suivante :

Le chapitre 1est destiné à fournir une vue d'ensemble ou nous éclaircirons ce que désigne le terme clustering. Nous donnons un aperçu des fondements théoriques de l'analyse des clusters et ce qui se fait dans ce domaine. Une brève introduction aux algorithmes du clustering implémentés dans cette thèse.

• Nous présentons dans le chapitre 2 des généralités sur les modèles d'exécution des programmes selon la taxonomie de Michael J. Flynnet ; un état de l'art sur les architectures et les plateformes parallèles les plus populaires auprès de la communauté des programmeurs parallèles, ainsi apporte également une vue génerale sur les différents langages de la programmation parallèle tel que ; MPI, OpenMP, Pthread, Intel TBB, et CUDA. De plus on présente les fondements de la programmation parallèle hétérogène basée sur le standard OpenCL tel que les modèles programmatiques, l'évolution et les projets majeurs échafaudés autour d'OpenCL.

• Le chapitre 3 présente la parallélisassions de l'algorithme du clustering K-means. On présente une étude comparative des quatre implémentations les plus performantes de l'algorithme du K-means : l'implémentation séquentielle de l'algorithme du Lloyd-Forgy, les implémentations parallèles ciblant le CPU en utilisant OpenMP / Pthreads et enfin une implémentation plus complexe qui utilise le Langage OpenCL. Un rapport détaillé sur l'implémentation OpenCL de cet algorithme, conçu au cours de cette recherche, est donné. Un résumé de l'état de l'art des implémentations est également présenté.

Le chapitre 4 suit le même format que le chapitre 3, et présente les recherches effectuées pour accélérer l'algorithme du K-means++. Une brève discussion de cette implémentation et une brève comparaison des résultats de synthèse seront données.

Le chapitre 5 concentre sur le développement d'une méthode de détection d'objets abandonnés dans la surveillance visuelle. La nouveauté de notre méthode est l'utilisation de l'algorithme du clustering parallèle k-means avant la soustraction du background ou chaque pixel est représenté par un cluster ensuite on utilise les contours au lieu des pixels dans la détection de régions statiques, pour la classification deux scores robustes sont également utilisés pour la classification des objets abandonnés [11], le temps d'exécution du K-mean traditionnel a été amélioré en proposons un algorithme parallèle kmeans en utilisant la plateforme OpenCL.

Enfin, on présente un résumé de la thèse et donne ainsi que des perspectives pour des futurs travaux connexes.

Références

[1] G. Moore, “Cramming more components onto integrated circuits,” Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. in Solid-State Circuits Society Newsletter, IEEE , vol.11, no.5, pp.33-35, Sept. 2006

[2] A. Huang, “The Death of Moore’s Law Will Spur Innovation,” Spectrum.ieee.org, 2015. [Online]. Available: http://spectrum.ieee.org/semiconductors/design/the-death-of-moores-law-will-spur-innovation.

[3] John Cheng, Max Grossman and Ty McKercher, professional CUDA C Programming, ISBN: 978-1-118-73932-7 Canada: wiley Brand, 2014.

[4] Benedict R Gaster et Lee Howes. Opencl c++. Dans GPGPU 6 Proceedings of 6th Workshop
on General Purpose Processor Using Graphics Processing Units. ACM, 2013.

[5] Benedict R. Gaster, Lee Howes, David R. Kaeli, Perhaad Mistry et Dana Schaa. Heterogeneous Computing with OpenCL, 2nd Edition Revised OpenCL 1.2 Edition, chaptitre DataManagement, pages 147–159. Elsevier Inc, Waltham USA, 2011.

[6] Aaftab Munshi, Benedict R. Gaster, Timothy G. Mattson, James Fung et Dan Ginsburg.OpenCL Programming Guide, chaptitre Programming with OpenCL C, pages 146–147.Adisson-Wesly, Upper Saddle River, NJ Boston , Indianapolis , San Francisco, 2011.

[7] Banger Ravishekhar et Bhattacharyya Koushik. OpenCL Programming By Example, pages
35–58. PACKT PUBLISHING, Birmingham UK, 2013.

[8] Tay Raymond. OpenCL Parallel Programming Development Cookbook., chaptitre Understanding how to solve SpMV using VexCL, pages 216–219. PACKT Publiching, Birmingham UK., 2013.

[9]H. Sarbazi-Azad, Advances in GPU Research and Practice, united states: Morgan, 2017.

[10] David Kirk et Wen-meiHwu. ProgrammingMassivelyParallel Processors, A hands-on Approach, second editon, USA: Morgan Kaufmann, Elsevier, Burlington USA, 2010.

[11] Ilias Dahi et al. “An edge-basedmethod for effective abandonedluggagedetection in complex surveillance videos”. In: Computer Vision and Image Understanding 158 (2017), pp. 141–151.

CHAPITRE1 :

Préliminaires : Les algorithmes de clustering

Sommaire

1.1 **Introduction**

Le Data Mining ou explorations de données est une composante essentielle des technologies Big Data. Le data mining est une technique qui consiste à extraire des connaissances à partir de données volumineuses, qui sont stockées dans des bases de données ou dans d'autres répertoires d'informations [1, 2]. Ces données volumineuses sont collectées chaque jour dans de nombreux domaines commerciaux et scientifiques. Les données provenant de ces répertoires sont utilisées dans les méthodes d'analyse prédictive pour extraire des résultats intéressants, ce qui aide à la prise de décision**.**

Le clustering ou l'analyse de cluster ou partitionnement de données en français, est une méthode importante dans Data Mining qui aide les analyseurs à comprendre le regroupement des attributs dans les données. Le clustering regroupe les objets de données en se basant uniquement sur les informations retrouvées dans les données qui décrivent les objets et leurs relations (voire **Figure 1.1 et Figure 1.2**). L'objectif est que les objets au sein d'un groupe soient similaires (ou liés) les uns aux autres et soient distincts (ou non liés) aux objets des

différents groupes, Cela correspond à la définition de [3]. Autre Définition [4] Le clustering est un processus qui regroupe un ensemble d'objets (physiques ou abstraits) en clusters similaires de telle sorte que les données du même cluster aient des caractéristiques similaires, et celles appartenant à des clusters distincts soient dissimilaires.

La qualité du clustering est déterminée par l'instinctivité de ces groupes, ainsi que par l'homogénéité au sein d'un même groupe [5, 6,7]. Idéalement, le degré de dissimilarité entre les sous-groupes devrait être plus important que le degré de similarité au sein de chaque groupe [8]. L'analyse des clusters est appliquée dans de nombreux domaines différents, allant des sciences naturelles (génotypage en biologie, prédiction des propriétés des molécules en chimie ou la classification des matériaux en géologie) à la sociologie (analyse des connections dans les réseaux sociaux) à l'informatique (vision par ordinateur, traitement multimédia, machine learning, data retrieval) et bien d'autres encore [3, 9,10].

Dans le domaine des images, cette description générale se traduit par les distances entre les pixels dans l'espace couleur [les pixels proches entre eux forment un groupe], et ils sont séparés si les distances sont relativement larges. Cependant, cette relativité contribue au problème majeur du clustering - les clusters de pixels sont très rarement séparés. Au lieu de cela, ils sont généralement connectés (c'est-à-dire que les distances des pixels " border " à deux ou plusieurs clusters sont similaires), et ils créent une sorte de nuages dont la densité change dans l'espace.

À ce jour, un certain nombre d'algorithmes du clustering ont été conçus et ils peuvent généralement être divisés en deux catégories principales : ***clustering hiérarchique*** et ***clustering partiels (***non hiérarchiques***)***. Clustering hiérarchique joue un rôle important dans Data Mining et l'imagerie, et il présente un avantage indéniable dans la mesure où le nombre de clusters n'a pas besoin d'être déterminé a priori. Sa nature linéaire entrave toutefois l'implémentation sur des systèmes à architecture parallèle, d'où la décision des auteurs d'orienter ses efforts vers la deuxième classe d'algorithmes - ***Clustering Partiel***, qui est en fait considéré comme adéquat dans la plupart des applications.Les algorithmes peuvent également être classés en fonction d'autres caractéristiques : exclusifs versus overlapping versus fuzzy, complets versus partiels. Bien que cette classification soit hors du champ de cette étude, il convient de mentionner que la version d'un algorithme étudiée plus loin dans cette thèse (K-means et k-means++) est exclusive et complète.

En premier lieu dans ce chapitre nous éclaircirons ce que désigne le terme clustering. Nous illustronsun aperçu des fondements théoriques de l'analyse des clusters et ce qui se fait dans ce domaine. Dans un second temps nous presenteront en détail de l'algorithme k-means, ainsi que de K-means ++ deux méthodes connues et efficaces auxquelles notre algorithme sera expérimentalement comparé.

1.2 **Problème de clustering**

Le clustering est l'affectation d'un ensemble d'observations à des sous-ensembles (appelés clusters) afin que les observations d'un même cluster soient similaires en quelque sorte. Le clustering est une méthode d'apprentissage non supervisée, et une technique commune d'analyse statistique des données utilisée dans de nombreux domaines, y compris machine learning, data mining, la reconnaissance des formes, les analyses des images et la bioinformatique. Leclustering est considéré comme étant adéquat dans la plupart des applications**.**

Le problème du clustering ou de l'analyse par clusters consiste à trouver des groupes d'objets à partir d'une grande collection de sorte que chaque objet d'un groupe soit plus semblable aux autres objets du groupe et différent des objets des autres groupes.

Le clustering utilise la distance entre les objets. La distance est généralement calculée comme une combinaison linéaire de la distance en fonction de plusieurs attributs. Ainsi, trouver une mesure de distance appropriée devient une étape importante du clustering. La mesure de la distance détermine la forme des clusters car deux objets peuvent être proches l'un de l'autre selon une dimension alors qu'ils seront éloignés selon une autre dimension. Nous choisissons la mesure de distance euclidienne pour notre thèse. Nous supposons également que toutes les caractéristiques ou toutes les dimensions ont la même pondération. Nos algorithmes et nos approches peuvent être généralisés de manière appropriée à d'autres mesures de distance.

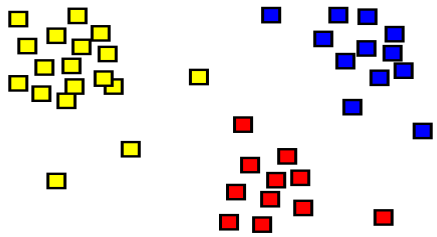


Figure 1-1: Résultat d'analyse par clusters représenté par la coloration des carrés en trois clusters.[35]

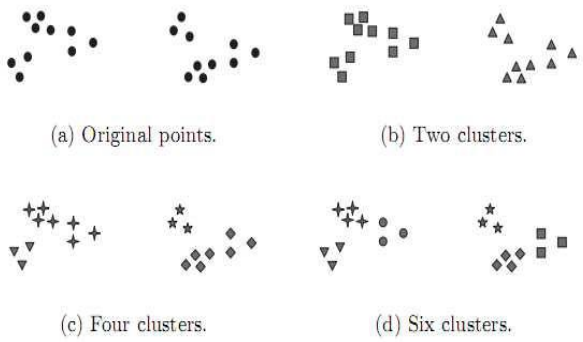


Figure 1-2: Différentes manières de clustering d'un même ensemble de points

Les méthodes du clustering hiérarchique[11] produisent une hiérarchie de clusters allant de petits clusters d'éléments très similaires à de larges clusters qui incluent des éléments plus dissimilaires.Elles peuvent être ascendantes ou descendantes. Les méthodes hiérarchiques produisent généralement une sortie graphique connue sous le nom de Dendrogramme ou (arbre hiérarchique, un exemple est donné dans la figure 1.3 ou la section horizontale indique à quel niveau la hiérarchie les subdivisions ne comptent plus,ce qui produit dans cet exemple trois partitions) qui montre cette structure de clustering hiérarchique [12, 9]. Les méthodes hiérarchiques descendantes sont divisantes, qui divisent progressivement large cluster comprenant toutes les données en deux clusters plus petits et répètent ce processus jusqu'à ce que tous les clusters aient été divisés. Les méthodes hiérarchiques ascendantes sont agglomératives et fonctionnent dans le sens inverse en trouvant d'abord les clusters des éléments les plus similaires et en ajoutant progressivement des éléments moins similaires jusqu'à ce que tous les éléments aient été inclus dans un seul grand cluster.

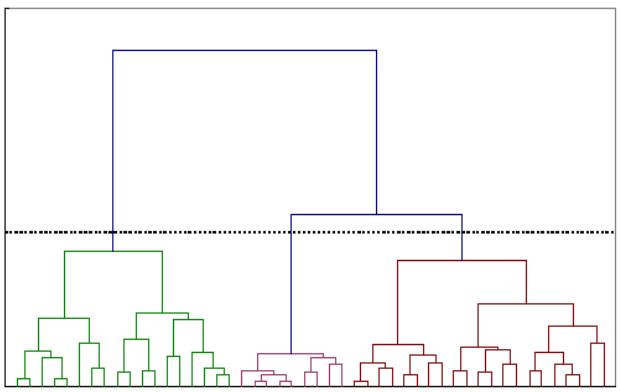


Figure 1-3: Exemple de hiérarchie sous forme standard de dendrogramme. [36]

Les méthodes hiérarchiques sont particulièrement utiles dans la mesure où elles ne sont pas limitées à un nombre prédéterminé de clusters et peuvent afficher une similarité des échantillons sur une large gamme d'échelles.

Les méthodes du clustering partiel (non hiérarchiques) :

Les méthodes de partitionnementdivisent l'ensemble de données en un certain nombre de groupes prédésignés par l'utilisateur (un exemple est donné figure 1.4).

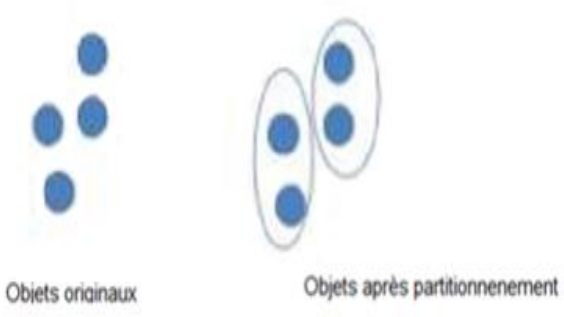


Figure 1-4: Exemple de clustering par partitionnement

Nous pouvons définir le partitionnement comme suit. Soit $X = \{x_1 , x_2 ,\dots , x_n\}$ représent un ensemble de n entités - pixels, et $P_k = \{C_1,C_2 , \dots,C_k\}$ être un ensemble de k clusters ($k < n$) contenant des entités/objets de X telles que :

- $Cj \neq Ø$; $j = 1,\dots., k.$
- $Ci \cap Cj = Ø$; $i; j = 1; : : : ; k$ et $i \neq j$; et $\bigcup_{j=1}^{k} C_j = X.$

Le clustering partiel est connu sous le nom de clustering de la somme des carrés minimum (MSSC) qui est formulé (**équation 1.1**) comme suit :

$$min_{x,c} \sum_{i=1}^{n} \sum_{j=1}^{k} s_{ij} \|x_i - c_j\|^2 \qquad (1.1)$$

Les propriétés du MSSC sont :

1. Elle exprime à la fois l'homogénéité et la distinction [13]
2. Les centres des clusters sont localisés dans leurs centroïdes (ou centres de gravité) ; un centroïde est défini comme suit (**équation 1.2**):

$$c_j = \frac{1}{m_j} \sum_{x \in c_j} x \qquad (1.2)$$

Où C_j est un centroïde (ou centre de gravité), x est une entité du cluster C_j et m_j est le nombre d'entités dans un cluster donné ;

3. Chaque entité est assignée à son centre de gravité le plus proche ;
4. Les clusters obtenus sont globaux.

clustering hiérarchique	**clustering partiel (non hiérarchique)**
Avantages du clustering hiérarchique : • Flexibilité incluse concernant le niveau de la granularité. • Facilite de manipuler toutes formes de similitude ou de distance. • Applicabilité `à tout type d'attribut. Inconvénients : • Imprécision sur les critères d'arrêt. • La plupart des algorithmes hiérarchique ne revisitent pas des classes (d'intermédiaire) une fois construits ils sont construits.	Avantages du clustering par partienement : • Permettent la classification d'ensembles volumineux. Inconvénients : • On impose au départ le nombre de classes.

1.3 **L'algorithme du K-means**

Les recherches sur K-means remontent au milieu du siècle dernier et ont été menées par de nombreux chercheurs de différentes disciplines, notamment Lloyd (1957, 1982) [14], Forgey (1965) [15], Friedman et Rubin (1967) [16] et MacQueen (1967) [17]. Jain et Dubes (1988) fournissent un historique détaillé de K-means ainsi que la description de différentes versions [12].L'algorithme du Lloyd, couramment appelé algorithme du K-means, l'algorithme du clustering K-means a été inventé pour la première fois par Stuart Lloyd en 1957, mais n'a été publié qu'en 1982 [14]. Indépendamment, a été proposé en [15]. Dans [18], il est désigné comme l'une des dix meilleures méthodes de data mining. L'objectif de cet algorithme est de diviser **N** points de **D** dimensions en **K** clusters afin de minimiser la somme aux carrés à l'intérieur du cluster. Le problème de k-means utilise une approche basée sur les centres pour le clustering partiel : chaque cluster est représenté par un centroïde(ou centre de gravité).

L'algorithme de clustering K-means implique le partitionnement itératif des données en k clusters (voire Figure 1.5). Il figure parmi les algorithmes de data mining le plus populaires [18], cet algorithme peut fournir de bons résultats dans de nombreuses situations pratiques et

il est utilisé dans de nombreuses autres applications telles que le traitement d'images, machine learning, la détection des anomalies [19] et la segmentation des données [20]. L'entrée de l'algorithme k-means est une matrice de **N** points de données, chacun ayant **D** dimensions et **K** centres de clusters initiaux [21].

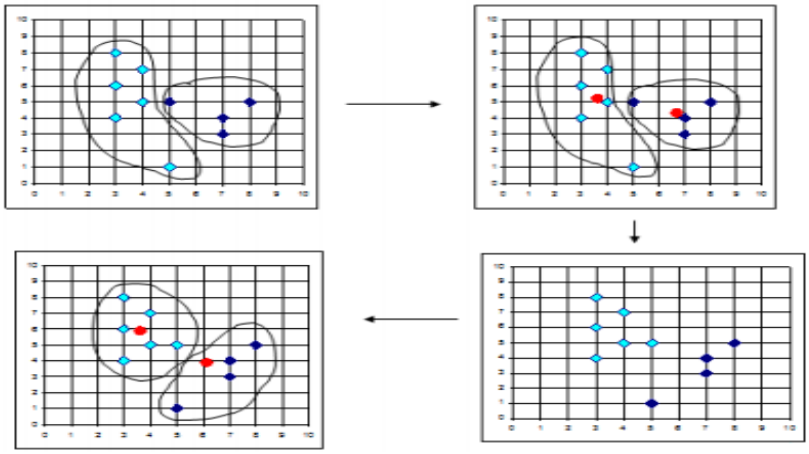


Figure 1-5: Exemple sur l'algorithme K-means

En statistique et en machine learning, clustering k-means est une méthode d'analyse de cluster qui vise à partitionner les **N** observations en **K** clusters dans lesquels chaque observation appartient au cluster ayant le moyen le plus proche.

Cependant, k-means est très long à mettre en œuvre lorsque les données ou le nombre des clusters est important. L'algorithme du clustering k-means fonctionne sur un ensemble de données de dimension **D**, $X = \{x_1, x_2, \ldots, x_n\}$ pour les partitionner en **K** clusters, où **N** est le nombre total de points de données.

Le résultat final est un ensemble de centres de gravité à D dimensions pour les clusters $C = \{c_1, c_2, \ldots, c_k\}$, ainsi qu'un ensemble de membres $M = \{ m_1, m_2, \ldots, m_n\}$ qui enregistre le cluster dont il est le plus proche pour chaque point de données. Un ensemble de centroïdes de clusters initiaux doit également être fourni. Il existe de nombreuses façons de déterminer les clusters initiaux. Selon la méthode utilisée, le centroïde résultant et la vitesse de convergence peuvent être très différents. La méthode la plus courante consiste à choisir au hasard k points de données comme clusters initiaux. Une méthode plus optimisée de sélection des clusters initiaux appelée k-means++ a été proposée [22], permettant une convergence plus rapide.

L'algorithme est constitué en deux étapes :

Étape d'affectation : Assigner ou attribuer chaque point de données au cluster de manière à minimiser la somme des carrés intra-cluster (WCSS) (ie variance). « within-cluster sum of squares (WCSS) » :

Où chaque X_n est affecté exactement à un cluster, même s'il pourrait être affecté à deux ou plusieurs d'entre eux (**équation 1.3**).

$$S_i^{(t)} = \{x_n : \left\| x_n - c_i^{(t)} \right\|^2 \leq \left\| x_n - c_{ji}^{(t)} \right\|^2, \forall j, 1 \leq j \leq k\} \tag{1.3}$$

Étape de mise à jour : Calculer les nouvelles moyennes qui seront les centroïdes des observations dans les nouveaux clusters (**équation 1.4**):

$$C_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x \in S_i^{(t)}} x_n \tag{1.4}$$

Le pseudo-code de l'algorithme (**Algorithme 1.1**) est :

Algorithme 1.1 : L'algorithme de base du K-means

1 : Sélectionner les K points aléatoirement, comme centroïdes initiaux

2 : Répéter

3 : Assigner chaque point à son centroïde le plus proche pour former les K clusters ;

4 : Recalculez le centroïde (centre de gravité) de chaque cluster ;

5 : **Jusqu'à ce qu'**aucun centroïde ne change.

1.3.1 Initialisation

L'initialisation des moyennes joue un rôle important sur les résultats en sortie. Si elles sont mal initialisées, on risque de faire converger l'algorithme sur un minimum local potentiellement éloigné du minimum global.

Une méthode basique (méthode de Forgy) d'initialiser les k moyennes initiales consiste à choisir des objets de données aléatoires, en utilisant la distribution uniforme, comme des centroïdes du cluster [17]. Des variantes de l'algorithme de base ont donc été proposées comme K-means++ [22], une méthode d'initialisation populaire, initialise les centroïdes de grappe par des objets choisis avec une probabilité proportionnelle au carré de la plus courte distance aux centroïdes déjà initialisés. Il a été démontré [22] que l'initialisation par K-means++ améliore à la fois l'efficacité et l'efficience. Une limitation majeure de K-means++ est dans sa nature séquentielle. Cependant, une version de la méthode, appelée K-means||, qui peut être efficacement parallélisée, a été proposée [23]. Une brève description des autres

méthodes d'initialisation et une comparaison expérimentale approfondie en termes d'efficacité et d'efficience sont disponibles dans [24]. Les moyennes peuvent également être remplacées par des médianes [25] ou encore par la somme de fonctions variées comme l'algorithme Fuzzy C-means [26] où l'appartenance n'est plus exclusive mais traduite par un degré d'appartenance.

Pour l'algorithme K-means, les méthodes d'initialisation les plus courantes sont Forgy et Random Partition. La méthode Forgy choisit de manière aléatoire les k pixels de données à partir de l'ensemble de données et les utilise comme moyennes initiales. La méthode de partitionnement aléatoire attribue d'abord un index aléatoire à chaque point de données, puis passe à l'étape de mise à jour en supposant que les points de données ayant le même index sont considérés comme étant attribués au même cluster, en calculant ainsi les moyennes qui seront les centroïdes des points attribués aléatoirement à chaque cluster. La méthode de Forgy tend à étaler les moyens initiaux, tandis que la méthode de Partition aléatoire les place tous près du centre de l'ensemble de données. Selon Hamerly et al, la méthode de la partition aléatoire est généralement préférable pour les algorithmes tels que k-harmonicmeans et fuzzy K-means. Pour les algorithmes maximization d'espérance et standard K-means, la méthode Forgy d'initialisation est préférable. Cependant, une étude approfondie de Celebi et al. ont trouvé que les méthodes d'initialisation populaires telles que Forgy, Random Partition et Maximin sont souvent peu performantes [24].

1.3.2 **La sélection de la distance**

Afin de mesurer la similarité ou la régularité entre les objets de données, la métrique de la distance joue un rôle très important. Il est nécessaire d'identifier de quelle manière les données sont liées entre elles, comment différentes données sont dissimilaires ou similaires entre elles et quelles mesures sont considérées pour leur comparaison. L'objectif principal du calcul de la métrique dans un problème spécifique est d'obtenir une fonction de distance/similarité appropriée. L'apprentissage métrique est devenu une question populaire dans de nombreuses tâches d'apprentissage et il peut également être appliqué dans une grande variété de contextes, puisque de nombreux problèmes d'apprentissage impliquent la notion définie la distance ou la similarité [5]. La fonction métrique ou fonction de distance est une fonction qui définit la distance entre les éléments/objets d'un ensemble [5,6]. Un ensemble avec une métrique est connu comme un espace métrique. Cette métrique de distance joue un rôle très important dans les techniques de clustering. De nombreuses méthodes sont disponibles pour le clustering.

En général, on définit une fonction Similarity(X,Y), où X et Y sont deux objets ou ensembles d'une certaine classe, et la valeur de la fonction représente le degré de "similarité" entre les deux. Formellement, une fonction de distance est une fonction **Dist** à valeurs réelles positives, définie sur le produit cartésien X xX d'un ensemble X. Elle est appelée une métrique de X si pour chaque x, y, z € X **:**

- Dist(x,y)=0 if x=y (l'axiome d'identité);
- Dist(x,y) + Dist(y,z) ≥ Dist(x,z) (l'axiome du triangle);
- Dist(x,y)=Dist(y,x) (l'axiome de symétrie).

L'espace métrique fournit un ensemble X.

1.3.2.1 **La distance euclidienne**

La distance euclidienne permet de calculer la racine carrée de la différence entre les dimensions d'une paire d'objets (**équation 1.5**).

$$Dist_{xy} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \tag{1.5}$$

1.3.2.2 **La distance de Manhattan**

La distance de Manhattan permet de calculer les différences absolues entre les dimensions d'une paire d'objets (**équation 1.6**).

$$Dist_{XY} = |X_{ik} - X_{jk}| \tag{1.6}$$

1.3.2.3 **La distance de Tchebychev**

La distance de Tchebychev (**équation 1.7**), également connue sous le nom de distance de valeur maximale, est calculée comme la magnitude absolue des différences entre les dimensions d'une paire d'objets.

$$Dist_{XY} = max_k|X_{ik} - X_{jk}| \tag{1.7}$$

1.3.2.4 **La distance de Minkowski**

La distance de Minkowski (**équation 1.8**) est la distance métrique généralisée.

$$Dist_{XY} = \left(\sum_{K=1}^{d} |X_{ik} - X_{jk}|^{\frac{1}{P}}\right)^{P} \tag{1.8}$$

Notez que lorsque p=2, la distance devient la distance euclidienne. Lorsque p=1, elle devient la distance Manhattan. La distance de Chebyshev est une variante de la distance de Minkowski où p=∞ (en prenant une limite). Cette distance peut être utilisée pour les variables ordinales et quantitatives.

K-means minimise la variance intra-cluster. La définition de la variance est identique à la somme au carré des distances euclidiennes du centre. L'idée de base de l'algorithme du K-means est de minimiser les erreurs quadratiques. La preuve commune de la convergence est la suivante : l'étape d'affectation et l'étape de mise à jour de la moyenne permettent toutes les deux d'optimiser le même critère. Par conséquent, elle doit converger après un nombre fini d'améliorations/ de modifications.

1.3.3 **La Convergence**

La convergence est définie comme une situation idéale dans laquelle il n'y a pas de réaffectation des pixels de données après un nombre fini d'itérations. Il faut donc définir une condition de terminaison. La condition de terminaison pourrait être l'une des suivantes

1. Un nombre fixe d'itérations est atteint. Cette condition limite le temps d'exécution de l'algorithme de clustering, mais dans certains cas, la qualité du clustering sera mauvaise en raison d'un nombre insuffisant d'itérations.
2. L'affectation des pixels de données aux clusters ne change pas entre les itérations. Sauf dans les cas où le minimum local est mauvais, cela produit une bonne clustering, mais le temps d'exécution peut être trop long.
3. Les centroïdes ne changent pas entre les itérations. En pratique, le changement des centroïdes entre les itérations peut être déterminé et nous le terminons souvent dès que le taux de changement ou la valeur absolue du changement est inférieur à un seuil donné. Cela est généralement associé à des situations où certains pixels de données peuvent être proches de deux centroïdes différents. La classification avec l'un ou l'autre cluster génère une très petite différence. Si l'algorithme du K-Means fonctionne pour un grand nombre d'itérations, les changements de centroïdes deviennent de plus en plus petits. Dans le pratique, la classification de ces pixels de données n'est pas très significative.
4. Terminer lorsque le WCSS est inférieur à un seuil. Ce critère permet de s'assurer que le clustering est d'une qualité souhaitée après la terminaison du l'algorithme. Il indique que la terminaison est proche de la convergence. Cette propriété peut être facilement vérifiée par sa définition de la somme des carrés à l'intérieur du cluster. Si la différence entre les itérations est plus petite au-dessus du seuil, la condition de terminaison peut également être atteinte.

Dans cette recherche, la première condition de terminaison est sélectionnée. Cela est dû aux raisons suivantes. La motivation première est d'explorer l'accélération de l'implémentation sur les GPUs. Si le nombre d'itérations est fixe, il est beaucoup plus facile de déterminer le facteur d'accélération. Deuxièmement, il est plus facile à implémenter cette terminaison sur le

matériel et de réduire l'utilisation de ressources matérielles supplémentaires. Enfin, il est préférable de débugger et de tester lorsque nous devons mettre fin à cet algorithme.

1.3.4 **La complexité de l'algorithme du K-means**

La complexité d'algorithme doit garder la trace des centroïdes de cluster ainsi que les éléments de données utilisés. Par conséquent, la **complexité spatiale** de l'algorithme est limitée par le nombre d'éléments de données **N** et le nombre de centroïdes de cluster **K**, chacun ayant une dimension donnée **D** [27] :

O((N + K)D)

En théorie, **la complexité temporelle** au pire des cas est déterminée par le nombre de configurations de clustering possibles (le nombre de cellules de Voronoï possibles) : Inaba et al. [28] ont trouvé que c'est en O(nkd). Arthur et al. [29] ont prouvé que la complexité de l'algorithme du k-Means est polynomiale. Bien que la complexité dans le pire des cas soit de O(nkd), le temps d'exécution est normalement polynomial [10]. Le temps d'exécution moyen de l'algorithme dépend de plusieurs facteurs, notamment le nombre d'éléments dans les données d'entrée **N**, la dimensionnalité **D**, le nombre de clusters **K** et le nombre d'itérations jusqu'à convergence **Iter** , donc : O(N*K*D*Iter)

Avantages :

- Rapide, robuste et plus facile à comprendre.
- Donne le meilleur résultat lorsque les ensembles de données sont distincts ou bien séparés les uns des autres.

Inconvénients :

- L'algorithme d'apprentissage nécessite une spécification à priori du nombre de centres de clusters.
- L'utilisation de l'affectation exclusive - S'il y a deux données qui se chevauchent fortement, alors k-means ne sera pas capable de résoudre qu'il existe deux clusters.
- L'algorithme d'apprentissage n'est pas invariant aux transformations non linéaires, c'est-à-dire qu'avec une autre représentation des données, on obtient des résultats différents (des données représentées sous forme de coordonnées cartésiennes et de coordonnées polaires donneront des résultats différents).
- La distance euclidienne peut pondérer de manière inégale les facteurs sous-jacents.

- L'algorithme d'apprentissage fournit les optima locaux de la fonction d'erreur quadratique.
- Le choix aléatoire du centre du cluster ne peut pas nous conduire à un résultat significatif.
- Cet algorithme ne fonctionne pas bien pour les données catégorielles, c'est-à-dire qu'il n'est applicable que lorsque la moyenne est définie.
- Il est incapable de traiter les données bruitées et les valeurs aberrantes.
- L'algorithme ne fonctionne pas pour les ensembles de données non linéaires.

1.4 L'algorithme du K-means++

Bien que le clustering k-means soit une méthode efficace pour le clustering de grandes quantités de données, les clusters résultants peuvent être loin d'être optimaux. L'algorithme k-means++ [22] a été développé pour améliorer la qualité des clusters obtenus par la sélection d'un ensemble de clusters initiaux à partir des N points de données qui, en moyenne fournissent une meilleure représentation du clutering qu'une sélection aléatoire. k-means++ est rapidement révélée être l'une des plus populaires, avec des applications dans des domaines différents tels que l'analyse des réseaux sociaux [30], le clustering des informations géographiques [31], la compression d'images [32] et microblogs [33].

L'algorithme du k-means++ utilise une approche de probabilité pour sélectionner les clusters à partir des données, où la probabilité de choisir un point de données particulier est basée sur la distance au carré de ce point de données par rapport à touts les clusters précédemment sélectionnés. En pratique, cela signifie que les cluters sélectionnés tendent à être des points de données qui sont très éloignés les uns des autres, sans forcément être les plus dispersés.
Dans l'algorithme standard K-means, les centroïdes initiaux sont choisis de manière aléatoire à partir de l'ensemble des données. Bien que cette approche soit simple et rapide, elle peut parfois donner des résultats qui sont loin d'être optimaux, puisqu'elle n'offre aucune garantie de précision. En 2007, Arthur et Vassilvitskii [22] ont proposé un algorithme dans lequel les centroïdes initiaux sont choisis avec une probabilité spécique, c'est-à-dire qu'ils sont choisis parmi les points de données dont la probabilité est proportionnelle à leur contribution au potentiel global qui est, en réalité, la somme des erreurs au carré. Cette méthode a donné de meilleurs résultats pour moins de nombres d'itérations que l'algorithme du k-means de base.
L'algorithme est décomposé comme suit (**Algorithme 1.2**) :

Algorithme 1.2 :L'algorithme du K-means++

1 : Choisir le premier centroïde de façon uniforme et aléatoire à partir de l'ensemble des données

2 : répéter

3 : Choisir le prochain centroïde, en sélectionnant un point parmi l'ensemble des données avec probabilité P (voir la formule 1.9)

4 : Recalculer le centroïde le plus proche pour chaque point de l'ensemble de données

5 : Jusqu'à ce que tous les k centroïdes sont choisis.

6 : Continuer en appliquant l'algorithme standard du K-means.

La probabilité de choisir un point comme centroïde est donnée par la formule ci-dessous :

$$P(x') = \frac{D(x')^2}{\sum_{x \in X} D(x)^2} \quad (1.9)$$

Où $D(x)^2$ est la distance au carré entre le point donné et son centroïde le plus proche.
Autrement dit, plus le point ce situe loin des centroïdes déjà trouvés, plus la probabilité de le sélectionner est élevée. Une telle pondération garantit que les centres de cluster seront répartis sur l'ensemble de l'espace de données. Alors que K-means++ améliore l'algorithme en sélectionnant "intelligemment" les centres initiaux de cluster et améliore la qualité de la solution, les deux autres méthodes fonctionnent de manière plus active, en ce sens qu'elles réduisent la quantité de calculs à chaque itération.

- ***Méthode des medoïdes :*** Le K-medoïde [34] est le point de données le plus approprié au sein d'un cluster qui le représente. La représentation par les K-medoïdes a deux avantages :
  - ✓ Elle ne dépend pas du type d'attribut c-a-d elle ne présente aucune limitation sur les types d'attributs
  - ✓ Le choix des medoïdes dépend de la concentration des points à l'intérieur d'un cluster, de ce fait. Une fois les medoïdes choisis, les clusters sont définis comme des sous-ensembles de points proches de leurs medoïdes respectifs, et la fonction objective est définie comme la distance moyenne ou une autre mesure de dissimilarite entre un point et son medoïde.
- Le fonctionnement objectif du k-médiane est assez similaire à celui de la k-medoids, sauf que la "distorsion" entre un point de données et le centroïde de son cluster est mesuré par la distance, plutôt que par le carré de la distance :

1.5 **Conclusion**

Dans ce chapitre, nous avons donné un bref aperçu du concept de base du clustering ainsi que le problème du clustering lié au calcul de la distance qui permet de déterminer la forme des clusters, Nous choisissons la mesure de distance euclidienne pour notre thèse, puis on a défini deux catégories principales : ***clustering hiérarchique*** et ***clustering partiel***. Parmi les méthodes non hiérarchiques ; k-means et k-means++, dans l'algorithme K-means, les centroïdes initiaux sont choisis de manière aléatoire à partir de l'ensemble des données. Bien que cette approche soit simple et rapide, elle peut parfois donner des résultats qui sont loin d'être optimaux. Alors dans k-means++ est proposé un algorithme dans lequel les centroïdes initiaux sont choisis avec une probabilité proportionnelle, qui donnelemeilleur résultat et moins d'itérations que l'algorithme du k-means de base. Cependant, k-means et k-means++ sont très long à mettre en œuvre lorsque les données ou le nombre des clusters est important.

Dans le chapitre 2, nous abordons desgénéralités sur les modèles d'exécution des programmes selon la taxonomiede Michael J. Flynnet apporte également une vue génerale sur les différents langages de la programmation parallèle. Nous présenterons les principes de base, sur lesquels l'écosystème s'articule ainsi les modèles programmatiques. Nous présenterons aussi les fondements de la programmation parallèle hétérogène basée sur le standard OpenCL. Nous examinerons également les différents fournisseurs de matériel et de logiciels OpenCL.

1.6 **References**

[1] Mohammed El Agha, Wesam M. Ashour, "Efficient and Fast Initialization Algorithm for K-means Clustering," I.J Intelligent Systems and Applications, vol.1, pp.21-31,2012.

[2] Zhou Haiyan, Zhao Jianyang "A new K-means Initial Cluster Class of the Center Selection Algorithm Based on the Great Graph," In communications in Information Science and Management Engineering, vol.2, pp.17- 20,2012.

[3] Rui Xu, Donald Wunsch, et al. "Survey of clustering algorithms". In: *Neural Networks, IEEE Transactions on* 16.3 (2005), pp. 645–678.

[4] J. HAN & M. KAMBER, Data mining : Concepts and techniques, Management Systems (The Morgan Kaufmann Series in Data Management Systems),MORGAN KAUFFMAN, ISBN 1-55860-901-6, 2006.

[5] Tan P.-N., Steinbach M., Kumar V.: Introduction to Data Mining,490-497, Addison-Wesley, 2006.

[6] M. Eirinaki and M. Vazirg annis, "Web Mining for Web Personalization," ACM Transactions on Internet Technology (TOIT), vol. 3, no 1, pp. 1-27, 2003.

[7] K. A. Abdul Nazeer, SD Madhu Kumar and M. P. Sebastian, "Enhancing the k- means clustering algorithm by using a O(n logn) heuristic m ethod for finding better initial centroids," in Second In ternational Conference on Emerging Applications of Infor rmation Technology, pp.261- 264,2011.

[8] Eric Backer and Anil K Jain. "A clustering performance measure based on fuzzy set decomposition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 1 (1981), pp. 66–75.

[9] T Hitendra Sarma, P Viswanath, and B EswaraReddy. "A hybrid approach to speed-up the k-means clustering method". In: *International Journal of Machine Learning and Cybernetics* 4.2 (2013), pp. 107–117.

[10] Jonathan Drake. "Faster k-means clustering." PhD thesis. 2013.

[11] Stephen C Johnson. "Hierarchical clustering schemes". In : Psychometrika 32.3 (1967), p. 241–254.

[12] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. isbn: 0-13-022278-X.

[13] Spath H.: Cluster analysis algorithm for data reduction and classification of objects, New York: John Wiley & Sons, 1980

[14] S. Lloyd, "Least squares quantization in PCM," IEEE Trans. Inf. Theory, vol. 28, no. 2, pp. 129–137, 1982.

[15] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," Biometrics, vol. 21, pp. 768–769, 1965.

[16] Friedman, H., Rubin, J.: On some invariant criteria for grouping data. J. Am. Stat. Assoc. 62, 1159–1178 (1967).

[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, 1967, pp. 281–297.

[18] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J.McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J.Hand, and D. Steinberg, "Top 10 algorithms in data mining," Knowl.Inf. Syst., vol. 14, no. 1, pp. 1–37, 2008.

[19] Sequeira K, Zaki M. ADMIT: anomaly-based data mining for intrusions. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2002.p. 386–395.

[20] Williams GJ, Huang Z. Mining the knowledge mine. In: Australian Joint Conference on Artificial Intelligence.Springer; 1997. p. 340–348

[21] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100–108, 1979. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2346830.

[22] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07), 2007, pp. 1027–1035.

[23] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," Proc. VLDB Endow., vol. 5, no. 7, pp. 622–633, 2012.

[24] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," Expert Syst. Appl., vol. 40, no. 1, pp. 200–210, 2013.

[25] Leonard Kaufman et Peter Rousseeuw. Clustering by means of medoids. North-Holland, 1987.

[26] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting

Compact Well-Separated Clusters". In : Journal of Cybernetics 3.3 (1973), p. 32–57. doi : 10 . 1080 / 01969727308546046. eprint : http : / / dx .doi .org/10.1080/01969727308546046. url : http://dx.doi.org/10.1080/01969727308546046.

[27] Bin Zhang, Meichun Hsu, and UmeshwarDayal. "K-harmonicmeans-a data clusteringalgorithm". In: *Hewlett-Packard LabsTechnical Report HPL-1999-124*(1999).

[28] Mary Inaba, NaokiKatoh, and Hiroshi Imai. "Applications of weightedVoronoidiagrams and randomization to variance-based k-clustering". In: *Proceedings of the tenthannual symposium on Computationalgeometry*. ACM. 1994, pp. 332– 339.

[29] David Arthur, Bodo Manthey, and H Roglin. "k-Means has polynomial smoothedcomplexity". In: *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*. IEEE. 2009, pp. 405–414.

[30] P. Velardi, R. Navigli, A. Cucchiarelli, and F. D'Antonio, "A new content-based model for social network analysis," in *IEEE International ConferenceSemanticComputing*, 2008.

[31] S. s. Lee, D. Won, and D. McLeod, "Tag-geotagcorrelation in social networks," in *Proceeding of the 2008 ACM Workshop on Search in Social Media*, 2008

[32] G. Karch, "GPU-basedacceleration of selectedclustering techniques," Master'sthesis, SilesianUniversity of Technology, Gliwice, 2010.

[33] D. Inouye, "Multiple post microblogsummarization," University of Colorado at Colorado Springs, Tech. Rep., 2010, research Final Report.

[34] P. RAI & S. SINGH, A survey of clustering techniques, International Journal of Computer Applications (0975 - 8887) Volume 7- No.12, October 2010.

[35] "https://en.wikipedia.org/wiki/Cluster_analysis"

[36] "https://www.mathworks.com/help/stats/linkage.html"

CHAPITRE2 :

Architectures et programmation parallèle hétérogène

Sommaire

2.1 **Introduction**

De nos jours, les environnements informatiques sont de plus en plus multiformes, exploitant les capacités d'une gamme de microprocesseurs multicœurs, unités centrales de traitement (CPU), processeur de signaux numériques (DSP), matériel reconfigurable (FPGA) et l'unité de traitement graphique (GPU). Présentés avec une telle hétérogénéité, la programmation efficace pour telles architectures pose un certain nombre de défis à la communauté des programmeurs [1].

De nombreuses recherches ont été menées au cours des dernières années sur le calcul parallèle, qui a été le principal sujet de la recherche de nombreuses universités. Le calcul parallèle utilise plusieurs processeurs ou ordinateurs travaillant ensemble sur un algorithme ou une tâche commune. En raison des contraintes liées à la mémoire disponible, aux performances d'une seule unité de calcul et à la nécessité de réaliser une tâche rapidement, divers Framework de calcul parallèle ont été définis. De nos jours, tous les ordinateurs sont parallèles, même les téléphones portables sont des plateformes multicœurs et chacun fonctionne avec le Framework de calcul parallèle de son choix. Selon la définition de Wikipédia, le calcul parallèle est une forme de calcul dans laquelle de nombreux calculs sont effectués simultanément, fonctionnant sur le principe que les grands problèmes peuvent souvent être divisés en plus petits, qui sont ensuite résolus simultanément (en parallèle).

Dans la plus parts des cas, le calcule parallèle est considéré comme un moyen de réduire le temps d'exécution des applications qui nécessitent une grande quantité de calcul. La conception et l'implémentation d'un algorithme parallèle est plus difficiles que celle d'un algorithme séquentiel.

Il existe de nombreux standards de programmation du calcul parallèle ou des API, tels qu'OpenMPI, OpenMP, Pthreads, etc. Ce chapitre est consacré à l'étude des environnements parallèles les plus répandus auprès de la communauté des programmeurs parallèles et des chercheurs spécialisés et à la programmation parallèle OpenCL. Dans ce chapitre, nous commencerons par une discussion sur la classification de Flynn, de manière plus simple, puis on entame les différents types de programmation parallèle. Nous présenterons les principes de base, sur lesquels l'écosystème s'articule, les modèles programmatiques. Nous présenterons aussi OpenCL avec ces différents composants. Nous examinerons également les différents fournisseurs de matériel et de logiciels OpenCL.

2.2 Modèles d'exécution :

2.2.1 Classification de Flynn

Les architectures parallèles et les algorithmes parallèles sont strictement liés entre eux, puisqu'une architecture parallèle nécessite un programme parallèle qui va l'exécuter. Depuis 1966[2], [3], Michael J. Flynn a créé le premier système de classification pour des architectures d'ordinateur de multiprocesseur, appelés "Taxonomie de Flynn". Cette classification est basée sur les flots de données et les flots d'instructions, ils proposent quatre types d'architectures différents [2] : SISD, SIMD, MISD et MIMD. La classification est montrée dans le Tableau 2-1 suivant :

		Flot de Données	
		Simple	Multiple
Flot d'Instructions	Simple	SISD (Von Neumann)	SIMD (Vectorielle et cellulaire)
	Multiple	MISD (pipeline)	MIMD (Multiprocesseur et passage de message)

Tableau 2-1: Classification de Flynn

2.2.1.1 SISD (instruction simple, une seule mémoire) :

Cette classe représente les ordinateurs séquentiels (l'architecture de Von Neumann), qui n'exploite aucun parallélisme. À chaque cycle d'horloge une seule instruction est exécutée sur un seul flot de donnée (figure2.1) ; ce type est caractérisé par sans déterminisme.

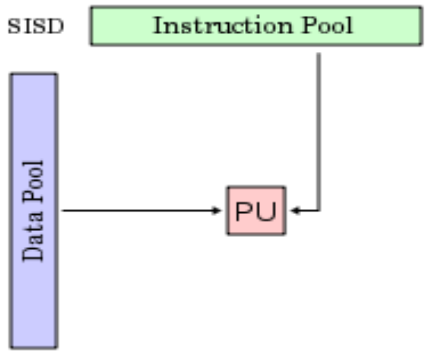


Figure 2-1: Architecture SISD [5]

2.2.1.2 SIMD (instruction simple, plusieurs mémoires)

Ce type représente une classe importante des architectures parallèle qui utilise le parallélisme au niveau de la mémoire, par exemple le processeur vectoriel où tous les processeurs

exécutent la même instruction à n'importe quel cycle d'horloge et chaque processeur peut traiter une donnée différente(Figure2.2). Ce type d'architecture est adapté aux problèmes ayant un degré élevé de régularité telle que le traitement d'image.

Cette architecture est caractérisée par son synchronisme et déterminisme.

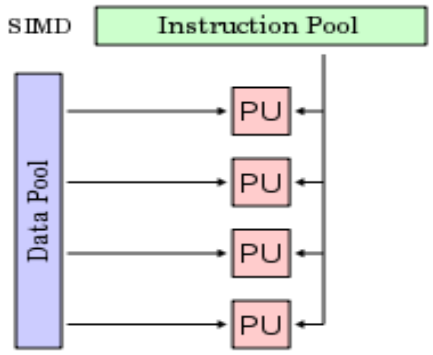


Figure 2-2: Architecture SIMD [5]

2.2.1.3 **MISD (instructions multiples, une seule mémoire)**

Il s'agit d'un ordinateur dans lequel une même donnée est traitée par plusieurs processeurs en parallèle. Le mode pipeline d'exploitation du parallélisme de contrôle correspond assez bien à cette classe. Il est du type MISD et correspond à un enchaînement de macro blocs, chacun ayant sa propre unité centrale (Figure2.3). Très peu de machines suivent cette architecture, elle peut être considérée lors de la création d'un modèle adapté à une situation particulière, par exemple de multiples transformations géométriques appliquées à un même polygone, filtrage numérique et la vérification de redondance dans les systèmes critiques.

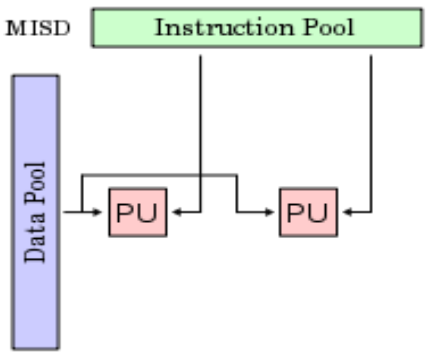


Figure 2-3: Architecture MISD [5]

2.2.1.4 **MIMD (instructions multiples, plusieurs mémoires)**

Le cas d'une machine MIMD ou cluster de multicœurs est le plus intuitif. Ici, plusieurs processeurs traitent des données différentes, car chacun d'eux possède une mémoire distincte, un flux différent d'instruction sur un flux différent de données, il s'agit de l'architecture

parallèle la plus utilisée (Figure 2.4). Ce type est le plus avancé et le plus répondu. La plus part des machines parallèles actuelles suivent cette architecture.

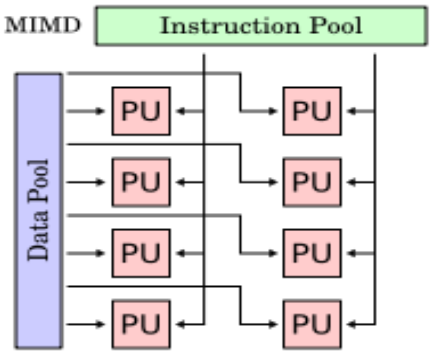


Figure :2-4: Architecture MIMD [5]

Nous rencontrons principalement les deux variantes suivantes.

- MIMD à mémoire partagée

Les processeurs ont accès à la mémoire comme un espace d'adressage global. Tout changement dans une case mémoire est vu par les autres CPU. La communication inter-CPU est effectuée via la mémoire globale (Figure 2.5).

- MIMD à mémoire distribuée

Chaque CPU a sa propre mémoire et son propre système d'exploitation. Ce second cas de figure2.6 nécessite un middleware pour la synchronisation et la communication.

Un système MIMD hybride est l'architecture la plus utilisée par les superordinateurs. Ces systèmes hybrides possèdent l'avantage d'être très extensibles, performants et à faible coût.

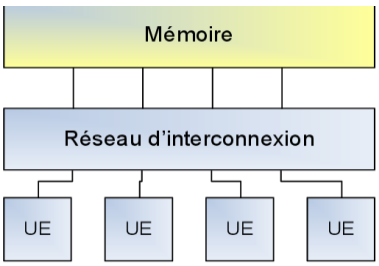


Figure 2-5: L'architecture de la mémoire partagée [22]

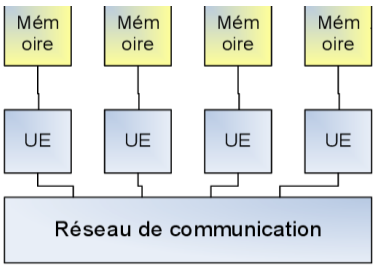


Figure 2-6: L'architecture de la mémoire distribuée [22].

Les processeurs qui sont basés sur l'architecture SIMD exploitent le parallélisme aux niveaux de données. Les mêmes instructions sont appliquées sur plusieurs morceaux de données en

même temps. Par contre, l'architecture MIMD est mieux adaptée au parallélisme au niveau des tâches.

2.2.2 **Classification de Duncan et Young**

Duncan [4] a proposé des modifications à la taxonomie de Flynn pour appréhender les nouveaux types des architectures selon les différentes méthodes d'accès physique à la mémoire [1, 6]. Selon sa classification il y a deux classes :

– Architectures à mémoire partagée ;

– Architectures à mémoire distribuée.

Les modèles à mémoire partagée donnent une vue de la mémoire selon un unique espace d'adressage : il s'agit de la vue qu'en ont les utilisateurs, et les applications qu'ils développent. Tandis que les modèles à mémoire distribuée, ils ne prévoient pas que la mémoire puisse être utilisée de façon partagée : c'est à dire chaque processeur dispose nécessairement de sa mémoire localement. En bas niveau, les échanges entre les processeurs (communication de données, synchronisation) doivent alors forcément s'effectuer par échange de messages [7, 8]. Le tableau 2.2 présente les avantages et les inconvénients de ces deux types d'architecture. En partant du type d'interconnexion, Young et al [9],proposent une classification plus fine des architectures selon l'accès physique à la mémoire :

	Mémoire partagée	Mémoire distribuée
Avantages	– l'espace d'adressage global permet un accès mémoire facile via un modèle de programmation intuitif – le partage des données se fait de manière rapide et uniforme grâce au rapprochement de la mémoire par rapport au processeur	– passage à l'échelle facile avec l'augmentation du nombre de processeurs – la taille mémoire augmente proportionnellement avec le nombre des processeurs – chaque processeur a accès à sa propre mémoire sans coût supplémentaire pour assurer la cohérence des caches mémoire.
Inconvénients	– passage à l'échelle difficile car l'ajout des processeurs peut augmenter géométriquement le trafic sur les interconnexions mémoire-processeurs – responsabilité de l'utilisateur pour faire la synchronisation – coût considérable pour la conception de machine avec un nombre conséquent des processeurs	– difficile à programmer : l'utilisateur doit se charger de toutes les communications inter-processeurs – temps d'accès non-uniforme – difficile de remodeler la répartition de données

Tableau 2-2: Avantages et inconvénients des architectures à mémoire partagée et distribuée

2.3 **Les langages parallèles**

2.3.1 **MPI (The Message Passing Interface):**

Le modèle de programmation MPI (Message Passing Interface) [10] est une spécification standard développée au milieu des années 90 pour les bibliothèques de sous-programmes pour la programmation parallèle de passage de messages qui sont portables sur des plates-formes informatiques parallèles à mémoire distribuée [11]. Il présente un avantage par rapport à OpenMP [14], car il peut exécuter sur l'architecture de mémoire partagée ou distribuée, mais OpenMP est utilisé comme solution de choix en mémoire partagée plus adaptée par rapport à l'interface MPI, car MPI demande un effort de programmation beaucoup plus important. Les ordinateurs à mémoire distribuée sont moins chers que les gros ordinateurs à mémoire partagée. Mais il a son propre inconvénient avec des difficultés inhérentes de programmation et de débogage. L'un des principaux inconvénients du Framework parallèle MPI est que les performances sont limitées par le réseau de communication entre les nœuds. Les superordinateurs ont un nombre massif de processeurs qui sont interconnectés par une connexion réseau à haut débit ou qui sont en clusters d'ordinateurs, où les processeurs sont très proches les uns des autres. Dans les clusters, il existe un bus de données dédié et coûteux pour les transferts de données entre les ordinateurs. Le MPI est largement utilisé dans la plupart de ces monstres de calcul appelés superordinateurs. Enfin, Framework MPI n'offre aucun support pour la programmation hétérogène.

Schéma général d'un programme MPI

Rang ← identificateur du processus en cours par rapport à un communicateur **Si** (rang = identificateur_spécifique) **Alors** faire un traitement **Sinon** faire un autre traitement

Il existe deux API standard pour la gestion des threads existants :
— PThreads[16] est une API commune aux OS(Linux et Unix) permet de fournir un ensemble de routines qui gère explicitement les threads ;
— OpenMP [15] est une API qui s'appuis sur l'ensemble des directives préprocesseur fournit une interface de plus haut niveau.

2.3.2 **Pthread :**

Mis en place à la fin des années 1980, POSIX fournit un ensemble de normes pour des interfaces de programmation de systèmes d'exploitation. Parmi celles-ci, la norme Pthreads [16] connu par Posix threads est un modèle, standardisée en 1995, permet de diviser un programme en sous-tâches dont l'exécution soit exécutée en parallèle. C'est une bibliothèque qui est utilisée de manière significative pour les applications multithread. Le multithreading est une technique qui permet à un programme d'effectuer plusieurs tâches simultanément. Le

standard Pthread[16] définit un ensemble des fonctions et de constantes pour le langage C permettant de contrôler la création, l'exécution et la synchronisation des threads [17]. Le programmeur est responsable de la gestion des threads et de la synchronisation. Cela offre plus de flexibilité et par conséquent de meilleures performances pour les programmes bien écrits [18].Dans les programmes Pthreads ; les variables globales sont partagées par tous les threads. Pour les variables locales, les arguments de fonction sont ordinairement privés au thread exécutant une fonction et si plusieurs threads exécutent la même fonction, chaque thread aura des copies privées de ces derniers parce que chacun a sa propre pile. Les programmes Pthreads sont généralement compilés et exécutés comme des programmes série et une façon relativement simple de spécifier le nombre des threads qui doivent être démarrés consiste à utiliser un argument de ligne de commande [19].

La richesse du Framework Pthread force à une restructuration globale des programmes souhaitant l'utiliser pour des besoins non triviaux. En cas où cette restructuration est trop coûteuse, l'utilisation du Framework OpenMP peut s'avérer plus adaptée.

2.3.3 **OpenMP**

OpenMP[12,13,14,15] (Open Multi-Processing) est une spécification de directives préprocesseur datant de 1997 et modèle de programmation qui prend en charge la programmation multiprocesseur en mémoire partagée multi-plateforme en C, C++ et Fortran. Elle n'est répandue que sur une plate-forme informatique multi-cœur avec un sous-système à mémoire partagée.

Voici un exemple de base de l'implémentation de la directive parallèle OpenMP :

```
#pragma ompparallel
   {
      Le corps;
   }
```

OpenMP permettent au développeur de modifier facilement le code existant pour l'exploiter l'architecture multicœur. OpenMP, bien qu'étant un excellent outil de programmation parallèle, ne permet pas l'exécution parallèle sur les périphériques hétérogènes, et l'utilisation d'une architecture multicœur avec sous-système de mémoire partagée ne la rend pas rentable.

2.3.4 **Intel TBB**

La bibliothèque Intel Threading Building sur des systèmes à un seul cœur du processeur, ainsi que sur des systèmes à plusieurs Blocks [20],[21],[23] développée par la société Intel prend en charge le parallélisme basé sur un modèle d'attribution des tâches.Lesprogrammes utilisant les Threading Building Blocks [51 ;52]sont exécutés cœurs du processeur. Threading Building Blocks permettent la programmation parallèle de données évolutives. En outre, il prend entièrement en charge le parallélisme imbriqué, de sorte qu'on peut facilement compiler de grands composants parallèles à partir de petits composants parallèles. Pour utiliser la bibliothèque, on doit spécifier des tâches, et non des threads, et laisser la bibliothèque mapper les tâches sur les threads de manière efficace. Le résultat est que Threading Building Blocks permet de spécifier le parallélisme de manière beaucoup plus pratique, et avec de meilleurs résultats, que l'utilisation de threads bruts.

Intel TBB présente les caractéristiques suivantes :

- C'est une bibliothèque supportant à la fois le parallélisme régulier et irrégulier.
- Des ordonnanceurs utilisés pour l'équilibrage des charges de travail.
- Permet d'utiliser des opérations atomiques et des allocations implicites de la mémoire.
- Permet d'utiliser des structures de données thread-safe (protégé des conditions de courses).
- Les modèles de parallélisme structure utilise (pipelines , fork–join, MAP, graphes de tâches, réduction).

Intel TBB est un DESL [21], et peut donc être utilisé avec tout compilateur supportant ISO C ++. TBB expose aux programmeurs un ensemble d'objets et de fonctions pour spécifier des blocs de code à exécuter en parallèle. A l'instar de la STL du C++ dans laquelle les interfaces sont spécifiées uniquement par des exigences sur les types de modèles, TBB suit une philosophie similaire. TBB repose donc sur des modèles et programmation générique avec peu d'hypothèses sur la nature des structures de données, ce qui maximise le potentiel de réutilisation des primitives TBB. TBB est basée sur une sémantique de tâches dans l'optique de réduire les frais généraux et de gérer plus efficacement les ressources. Les composants individuels de TBB peuvent également être utilisés avec d'autres modèles de programmation parallèle. Par exemple, Il n'est pas rare de voir l'allocateur de mémoire parallèle TBB utilisé avec des programmes OpenMP .

Presque tous les ordinateurs de bureau sont équipés d'un processeur multicœur et d'un GPU. Nous avons donc besoin d'un environnement de programmation dans lequel un programmeur peut écrire des programmes et les exécuter soit sur un GPU ou un CPU

multicœur. Alors que les CPU sont conçus pour gérer des tâches complexes, telles que le découpage du temps, les ramifications, etc., les GPU ne font bien qu'une chose ; ils gèrent des milliards d'opérations arithmétiques répétitives de bas niveau. Des langages de haut niveau, tels que CUDA et OpenCL, qui ciblent directement les GPU, sont disponibles aujourd'hui, de sorte la programmation GPU devient rapidement l'un des courants dominants de la communauté informatique.

Le tableau 2.3 résume le bilan de comparaison entre différents modèles de programmations parallèles avec différents accélérateurs. Parmi ces modèle de programmations parallèles OpenCL est le plus polyvalent est utilisé par différents accélérateurs.

	Cluster	Node CPU	Node GPU	Node Nvidia	Accelerator
MPI	Yes	Yes	No	No	Yes*
PVM	Yes	Yes	No	No	Yes*
OpenMP	No	Yes	No	No	Yes*
Pthreads	No	Yes	No	No	Yes*
OpenCL	No	Yes	Yes	Yes	Yes
CUDA	No	No	No	Yes	No
TBB	No	Yes	No	No	Yes*

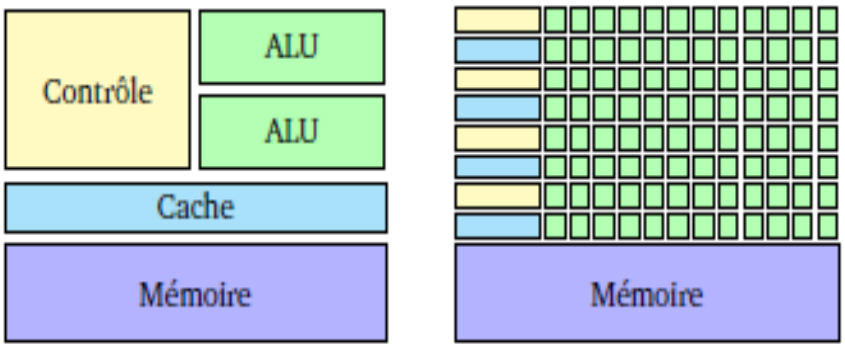
Tableau 2-3: Bilan de comparaison entre différents modèles de programmations parallèles avec différents accélérateurs.

2.4 Langages de programmation spécifiques aux accélérateurs graphiques GPU

Les GPU étaient difficiles à programmer car il fallait utiliser les API des fonctions graphiques (OpenGL [54]. et Direct3D [53]) ce qui limitait les performances et le type d'application parallélisée. Il existe deux Framework communs que les développeurs peuvent utiliser lorsqu'ils écrivent et exécutent des programmes dans des environnements informatiques hétérogènes. Tout d'abord, CUDA, qui a été introduit par NVIDIA en 2006, fournit au développeur une interface plus conviviale pour le GPU. Il s'agit d'une plate-forme de calcul parallèle et d'un modèle de programmation à usage général dans lequel les problèmes peuvent être décrits facilement et les calculs complexes peuvent être effectués avec efficacité [24]. Toutefois, il ne peut fonctionner que sur les appareils NVIDIA compatibles CUDA. Deuxièmement, OpenCL est une plate-forme de programmation parallèle à usage général et à code source libre, dans laquelle les développeurs peuvent développer des applications portables qui peuvent fonctionner sur divers processeurs multi-cœurs CPU et processeurs vectoriels tels que les GPU. Il apporte des abstractions substantielles pour réduire la complexité de programmation dans les environnements hétérogènes [25].

OpenCL est développé par le Groupe Khronos qui est un consortium industriel comprenant des leaders du secteur tels qu'AMD, NVIDIA, Apple et Intel. Fang et al [26] donnent une comparaison complète des frameworks CUDA et OpenCL. Ils concluent que la portabilité d'OpenCL ne réduit pas ses performances et qu'il peut être une bonne alternative à CUDA. OpenCL n'est pas moins performant que CUDA dans le cadre d'une comparaison équitable. Dans cette thèse, nous utilisons OpenCL version 1.2.

Les architectures CPU et GPU sont fondamentalement différentes l'une de l'autre. Les CPU modernes comprennent de nombreux cœurs et effectuent des traitements parallèles en implémentant l'architecture MIMD.La figure 2.7 illustre lacomparaisondes architectures entre un processeur central et un processeur graphique.



(a) Processeur central (b) Processeur graphique

Figure 2-7: Comparaison des architectures entre d'un processeur central et d'un processeur Graphique

La majeure partie de leur structure est constituée de l'unité de contrôle et du cache. Au contraire, les GPU se composent principalement de l'unité arithmétique et logique et réservent un espace minimal pour les unités de contrôle et de cache. En conséquence de ces différences d'architecture, il existe un compromis entre la flexibilité et la puissance de calcul. Les CPU ont été adaptés à la puissance de calcul tout en offrant des fonctionnalités générales performantes et les GPU sont plus efficaces pour effectuer de nombreuses opérations arithmétiques simultanément [27].OpenCL permet de traîner une partie de la puissance des GPUs, des CPUs multi-coeurs ou d'autres systèmes de calcul intensifs comme le CELL d'IBM ou des processeurs intégrés many-core Xeon Phi Intel via une seule infrastructure de programmation.

2.4.1 **Le framework CUDA – NVIDIA**

Les premiers GPU ont été conçus comme des accélérateurs graphiques, devenant plus programmables au cours des années 90, aboutissant au premier GPU de NVIDIA en 1999. Puis NVIDIA prend en charge une grande variété de langages de programmation qui améliorent le nombre et la portée des applications qui peuvent exploiter l'informatique parallèle sur le GPU.

Depuis sa création, l'écosystème CUDA s'est développé rapidement pour inclure des outils de développement logiciel, des services et des solutions basées sur les partenaires [28].
CUDA est une plate-forme informatique parallèle à usage général et un modèle de programmation qui tire partie du moteur de calcul parallèle dans les GPU NVIDIA pour résoudre de nombreux problèmes de calcul complexes d'une manière plus efficace.
On utilise CUDA, afin de pouvoir accéder au GPU pour le calcul, comme cela était traditionnellement fait sur le CPU. La plate-forme CUDA est accessible via ses bibliothèques accélérées, des directives de compilation, des interfaces de programmation d'application et des extensions de langages de programmation standard, notamment C, C ++, Fortran, Python et MATLAB et exprime le parallélisme par le biais d'extensions sous la forme de quelques mots clés de base[28][29].La programmation en CUDA repose sur deux modèles :

1. Modèle logique qui permet d'exprimer :
    - Le calcul à réaliser (Code C/C++)
    - L'organisation des threads sous forme d'une grille de blocs de threads
2. Modèle physique qui se charge de répartir et exécuter les threads sur les cœurs d'exécution du GPU [29].

CUDA est également un modèle de programmation évolutif qui permet aux programmes d'évoluer leur parallélisme avec les GPU de manière transparente[1].Elle fournit deux niveaux d'API pour gérer le périphérique GPU et organiser les threads, comme indique à la figure 2.8 ci-dessous :

1. API de pilote (Driver) CUDA : est une API de bas niveau et relativement difficile à programmer, mais elle offre plus de contrôle sur la façon dont le périphérique GPU est utilisé.
2. API d'exécution (Runtime) CUDA : est une API de niveau supérieur implémentée au-dessus de l'API du pilote. Chaque fonction de l'API d'exécution est décomposée en opérations plus simples émises pour l'API du pilote [29].

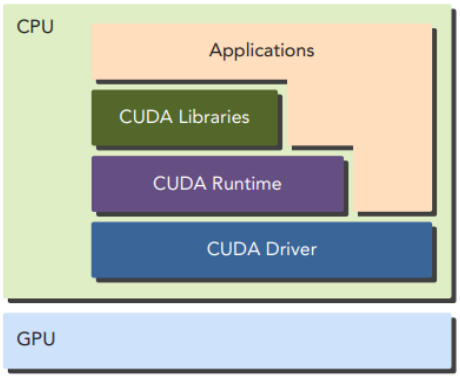


Figure 2-8: Les API sur CUDA [29]

La structure d'un programme CUDA reflète la coexistence d'un hôte (CPU) et d'un ou plusieurs périphériques (GPU) dans l'ordinateur. Chaque fichier source CUDA peut avoir un mélange de code d'hôte et de périphérique. Par défaut, une structure de programme CUDA typique comprend cinq étapes principales :

1. Allouer les mémoires GPU.
2. Copier les données de la mémoire CPU vers la mémoire GPU.
3. Appeler le noyau CUDA pour effectuer un calcul spécifique au programme.
4. Copier les données de la mémoire GPU vers la mémoire CPU.
5. Détruiser les mémoires GPU [29].

2.4.2 C++ AMP

C++ AMP (C++ Accelerated Massive Parallelism) [56, 57] permet d'accélérer l'exécution du code C++ en tirant l'avantage des données parallèles au matériel, tel qu'une unité de traitement graphique (GPU). En utilisant l'AMP C++, on doit coder des algorithmes de données multidimensionnelles de sorte que l'exécution peut être accélérée en utilisant le parallélisme sur des matériels hétérogènes. Le modèle de programmation C++ AMP comprend des tableaux multidimensionnels, l'indexation, le transfert de mémoire, le Tiling et la bibliothèque des fonctions mathématiques. Afin d'améliorer les performances, on peut utiliser les extensions du langage C++ AMP pour contrôler la façon dont les données sont déplacées du CPU au GPU et vice-versa.Le modèle de programmation C ++ AMP [55–57] est pris en compte dans les fichiers d'en-tête suivants :

• <amp.h>
• <ampgraphics.h>
• <amp court vectors.h>
• <arm_math.h>

2.4.3 **OpenACC**

OpenACC [58, 59] est une interface de programmation API récente pour la programmation d'accélérateurs en C/C++ et Fortran développée par Cray, CAPS, NVIDIA et PGI. L'API OpenACC pour C/C++ et Fortran, basée sur des directives, permet au compilateur de prendre en charge les tâches de programmation GPU de bas niveau, tout en assurant la portabilité entre les systèmes d'exploitation, les CPU hôtes et les accélérateurs. En ce qui concerne les types d'accélérateurs, jusqu'à présent, les implémentations OpenACC existantes ne prennent en charge que les GPU NVIDIA. Dans ce paragraphe, nous donnons un bref aperçu d'OpenACC, indiquons les cas d'utilisation les plus importants en ce qui concerne les GPU et faisons correspondre la terminologie à la nomenclature OpenCL. L'API OpenACC repose sur un modèle d'exécution dirigé par l'hôte dans lequel le programme principal s'exécute sur l'hôte et les zones de calcul intensif sont déchargées sur un accélérateur attaché. Le modèle de mémoire est basé sur la séparation des mémoires de l'hôte et des périphériques qui ne se synchronisent pas automatiquement. Les dispositifs GPU mettent en œuvre une modèle de mémoire faible cohérence entre les opérations et différentes unités de calcul et permet la cohérence au sein de la même unité de calcul uniquement en utilisant une synchronisation explicite.

En plus des directives, l'API OpenACC fournit également des appels de bibliothèque d'exécution et des variables d'environnement. Par exemple, les appels de bibliothèque peuvent recueillir des informations sur l'appareil, l'initialiser ou attribuer des données sur l'appareil.

2.4.4 **La plateforme DirectCompute de Microsoft**

Microsoft DirectCompute est une interface de programmation d'application (API) qui prend en charge l'exécution de noyaux de calcul sur des GPUs dans Windows Vista, Windows 7 et les versions ultérieures de Microsoft.

DirectCompute fait partie de la collection d'API Microsoft DirectX et a été initialement publié avec l'API DirectX 11 mais fonctionne sur des unités de traitement graphique qui utilisent DirectX 10 ou DirectX 11. Son architecture partage une gamme d'interfaces de calcul avec OpenCL à partir de Khronos Group, nuance de calcul en OpenGL et CUDA de NVIDIA[28].

L'API DirectX était un énorme bondissement en avant pour tous ceux qui se souviennent d'avoir programmé des cartes vidéo avant. Cela signifiait que les développeurs devaient apprendre une seule API de bibliothèque pour programmer toutes les cartes graphiques, plutôt

que d'écrire ou de faire une licence pilotes pour chaque grand fabricant de cartes vidéo. Il est assez facile de porter une application CUDA sur DirectCompute si on les maitrise [18].

2.4.5 **Plateforme Métal d'Apple**

Metal est le nom d'une nouvelle API graphique créée par Apple à destination des développeurs de jeux vidéo sur leurs appareils. S'appuyant sur une architecture abordable et à faible surcharge avec des shaders GPU précompilés, un contrôle des ressources à granularité fine et une prise en charge multithreading, cette API est beaucoup plus proche du système car elle est entièrement développée pour iOS et Mac OSX[30,31,32]. Ses performances pour les applications peuvent atteindre jusqu'à 8x celles de l'OpenGL.

Metal développe d'avantage la prise en charge de la création de commandes pilotée par GPU, simplifie le travail avec la gamme de GPU [30]. Elle a des fonctions très bas niveau, il est donc important de comprendre le fonctionnement des appels machine de l'API qui est séparée en deux librairies :

1. Metal : permet d'accéder directement au contrôle du GPU
2. MealKit. : fournit des utilitaires permettant de simplifier la création d'applications avec Metal.

Lors du développement direct sur GPU avec Metal, il est commun d'utiliser un "**MTLDevice**". Ce dernier représente un GPU, c'est à lui que l'on envoie des requêtes à l'aide d'un **MTLCommandQueue**, qui va s'occuper de transmettre les informations au GPU avec la méthode appropriée. Cette file d'attente contient des **MTLCommandBuffer** comportant eux-mêmes des **MTLCommandEncoder**, qui s'occupent de transformer le code en code interprétable par toutes les cartes graphiques et grâce à MetalKit, permettant d'utiliser des fonctions évitant de manipuler directement la **CommandQueue.** C'est devenu plus simple de créer du contenu 3D et MetalKit, permet d'utiliser des fonctions évitant de manipuler directement la CommandQueue et c'est devenu plus simple de créer du contenu 3D.
Metal est sur sa version 2, avec un grand nombre de nouveautés [32], son arrivée sur le marché a provoqué de nombreux changements, en bien comme en mal [31],le tableau 2.4 présente les avantages et les inconvénients de l'API Metal :

Les avantages	Les inconvénients
○ Interface très bas niveau avec le GPU, grâce à la maîtrise des périphériques utilisés sur les machines Apple. ○ Une librairie unique entre iOS et Mac OS X ○ Des grands acteurs du jeu vidéo derrière l'initiative: Autodesk, Unreal Engine, Unity Engine etc. ○ Un gain de performance ○ La possibilité de l'utilisation pour le Deep Learning, important dans le monde des intelligences artificielles. ○ Des calculs de shaders optimisés ○ Intégration poussée pour le développement avec XCode.	○ L'API a beaucoup de détracteurs car Apple reste toujours fermé malgré l'arrivée du langage Swift ○ Le développement n'est possible que sur des machines Apple tournant sur Mac OS X. ○ Metal signifie la fin du support d'OpenGL par Apple, ce qui peut signer la mort de cette API vieillissante. ○ Apple freine aussi le développement de la nouvelle API Vulkan (open-source et multi-plateforme).

Tableau 2-4 : Avantages et inconvénients de l'API Metal

2.5 La programmation des systèmes parallèle hétérogène « Le framework OpenCL »

Tout d'abord promu par Apple au début de 2008, OpenCL[46 ;47 ;48 ;49 ;50] a rapidement été soutenu par de nombreux autres vendeurs tels qu'IBM, Nvidia, AMD et Intel. Il fournit un ensemble de logiciels qui permet de résoudre les problèmes liés à la programmation de plates-formes hétérogènes de traitement parallèle .Aujourd'hui, OpenCL supporte les CPU qui incluent les x86, ARM et PowerPC et les GPU d'AMD, Intel et NVIDIA mais peuvent même être utilisés pour accélérer le traitement OpenGL ou Direct3D [33].Le langage OpenCL(Open ComputingLanguage) est un framework de programmation hétérogène qui est géré par leconsortium Khronos Group.Il est multiplateforme et largement supporté par les responsables de l'industrie [34].

OpenCL fournie une API en langage C et offre de nombreuses abstractions pour les routines matérielles de bas niveau, la gestion de la mémoire et les modèles d'exécution [35]. Le vocabulaire introduit par les spécifications d'OpenCL [25] est résumé ci-dessous : « **Host**: le CPU qui permet de coordonner l'exécution.

Device: c'est lui qui permet exécuter le code OpenCL C.

Kernel: Fonctions exécutées au sein du DeviceOpenCL.

Context:Création d'un contexte de travail dans lequel les éléments de travail s'exécutent ... inclut les périphériques et leurs mémoires et leurs files d'exécution.

Platform Model: Elle spécifie qu'il existe un processeur hôte coordonnant l'exécution, et un ou plusieurs périphériques dont le rôle est d'exécuter les noyaux OpenCL C. Il définit également un modèle abstrait du matériel pour les périphériques.

Execution Model: Il définit la façon dont l'environnement OpenCL est configuré par l'hôte, et comment l'hôte peut diriger les périphériques pour effectuer les tâches. Cela inclut la définition d'un environnement d'exécution sur l'hôte, des mécanismes d'interaction hôte-périphérique et un modèle de concurrence utilisé lors de la configuration des noyaux. Le modèle de concurrence définit comment un algorithme est décomposé en éléments de travail et en groupes de travail sous OpenCL.

Memory Model: Il définit les types de mémoire et la hiérarchie de mémoire abstraite que les noyaux utilisent indépendamment de l'architecture de mémoire sous-jacente actuelle. Elle contient également les règles d'ordonnancement de la mémoire et la mémoire virtuelle partagée facultative entre l'hôte et les périphériques.

Programming Model: Permet de définir comment le modèle de concurrence est mappé sur le hardware physique. »

2.5.1 **Le modèle OpenCL**

2.5.1.1 **Modèle de plate-forme**

La figure 2.9 présente un aperçu de l'architecture OpenCL.

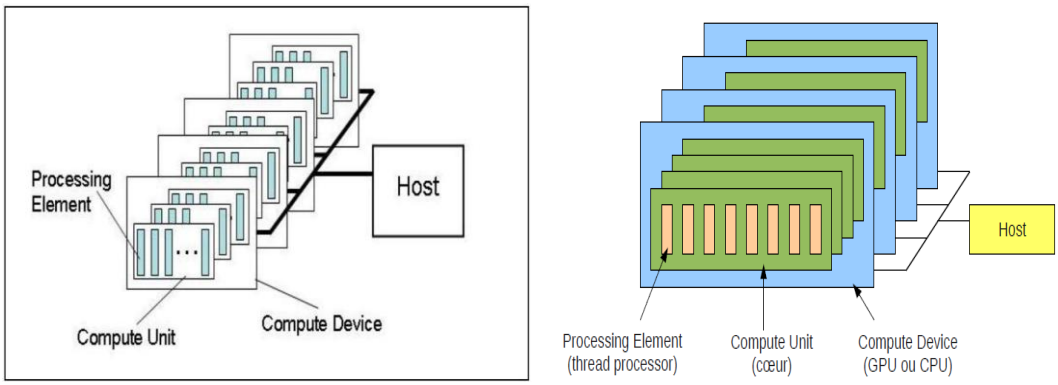


Figure 2-9: Modèle de plate-forme d'OpenCL [25].

Un programme OpenCL est constitué d'un hôte connecté à un ou plusieurs appareils OpenCL [25]. Le programme hôte est basé sur le CPU, et il contrôle l'exécution effectuée sur des appareils de calcul (ComputeDevices) qui peuvent être des CPU ou des GPU ou certains

autres appareils spécifiques. Ces appareils de calcul (ComputeDevices) sont constitués des unités de calcul (ComputeUnits), et les unités de calcul contiennent un ou plusieurs éléments de traitement(ProcessingElements) dans lesquels les instructions sont exécutées.

2.5.1.2 **Modèle d'exécution**

Un programme OpenCL fonctionne simultanément sur deux parties différentes. Un ou plusieurs appareils OpenCL exécutent les noyaux tandis que le programme hôte s'exécute sur l'hôte. L'application hôte charge le noyau sur le deviceOpenCL et déduit un espace d'index, appelé NDrange. Le device exécute une instance du noyau à chaque point de l'espace d'index. Ces instances sont appelées les éléments de travail (Work-Items) et sont identifiées par leurs IDs globale respectifs. Groupes de travail (Work-Groups) consistent en des éléments de travail (Work-Items) qui peuvent coopérer grâce à une mémoire commune et des barrières s'ils appartiennent au même groupe de travail (Work-Group). L'espace de l'index peut être à 1, 2 ou 3 dimensions. La figure 2.10 explique la structure d'un espace d'indexation à deux dimensions [25].

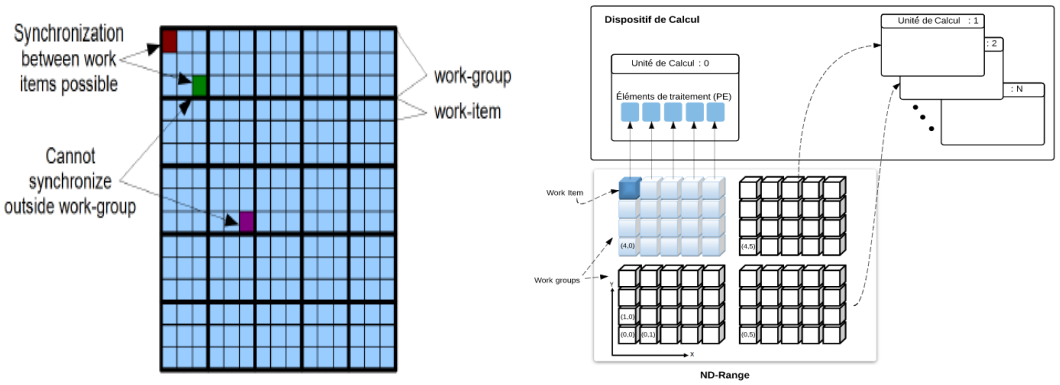


Figure 2-10: Structure de l'espace d'index à deux dimensions, extrait de [35].

2.5.1.3 **Modèle de mémoire**

Il existe quatre différentes modèles de mémoire auxquelles les éléments de travail (Work-Items) peuvent accéder. Les spécifications d'OpenCL les dénomment comme suit.

Mémoire globale : Ce modèle de mémoire pouvant être accessible par tous les éléments de travail avec accès en lecture/écriture.

Mémoire constante : Ce modèle de mémoire pouvant être accessible par tous les éléments de travail avec un accès en lecture seule.

Mémoire locale : Ce modèle de mémoire peut être accessible par les éléments de travail du même groupe de travail.

Mémoire privée : Ce modèle de mémoire privée est rèservè pour un élément de travail.

Les données doivent être placées dans l'une de ces quatre modèles de mémoire et l'emplacement des données influe sur la rapidité avec laquelle on peut y accéder. Un noyau qui accéde à la mémoire locale et plus rapide qu'un noyau qui accéde à la mémoire globale. Tant qu'un espace mémoire unifié est utilisé, le transfert de mémoire entre l'hôte et les appareils (Devices) doit être explicitement contrôlé. Le programmeur est responsable du déplacement des données de l'hôte vers l'appareil (Device) et vice versa. La plupart du temps, l'hôte et les appareils ont des espaces mémoire différents. La figure 2.11 montre les modèles de la mémoire.

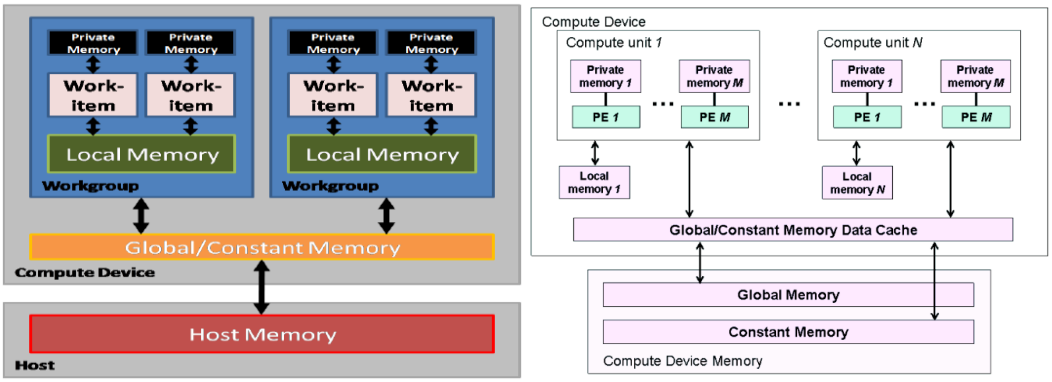


Figure 2-11: Modèle de mémoire d'OpenCL [36]

2.5.1.4 **Modèle de programmation**

Le modèle de programmation d'OpenCL prend en charge les modèles de programmation parallèle des données et des tâches. En outre, il est possible d'utiliser un hybride de ces deux approches. Notre implémentation est basée sur le modèle de données parallèles qui permet au modèle d'exécution de gérer la façon dont les données sont mappées sur les éléments de travail. Une instance d'un noyau fonctionne sur la partie spécifiée des données. Un exemple d'introduction extrait du guide de programmation OpenCL d'AMD [36] démontre la différence conceptuelle entre les approches scalaires et parallèles de données. La fonction scalaire donnée dans la liste 2.1 fonctionne de manière itérative à travers la boucle for. D'autre part, le noyau OpenCL parallèle de données indiqué dans la liste 2.2 ne lit que les éléments correspondants de l'entrée et écrit le résultat dans la partie correspondante de la sortie. La fonction est appelée pour chaque work item.

Liste 2.1 : Un exemple de fonctionnement scalaire en C

```
1 void square(int n, const float *a, float *result)
2 {
3    int i;
4    for (i=0; i<n; i++){
5    result[i] = a[i]*a[i];
6     }
7 }
```

Liste 2.2 : Un exemple de noyau OpenCL parallèle de données

```
1 kernel void dp_square
2       (global const float*, global float *result)
3  {
4       int id = get_global_id(0);
5       result[id] = a[id]*a[id];
6  }
7 // dp_square executes sur "n" work-items
```

2.5.2 Évolution des standards OpenCL

2.5.2.1 Le standard OpenCL 1.0

OpenCL 1.0 est la première version, en 2008 group Khronos a annoncé la ratification et la publication de la spécification d'OpenCL 1.0, c'est la première norme ouverte et sans redevance pour la programmation parallèle multiplateforme de processeurs modernes[37, 38].

2.5.2.2 Le standard OpenCL 1.1

Cette nouvelle version est mise par le groupe Khronos le 14 juin 2010 pour avoir des améliorations des performances et résoudre des problèmes portés par la première version. Les modifications ont touché les API OpenCL et OpenCL C[37 ,38 ,39]:

- Rajout de nouveaux types de données vectorielles à 3 composants.
- Formats d'image optionnels supplémentaires.
- Prise en charge de la capacité des objets de sous-tampons à créer un objet tampon qui fait référence à une région spécifique dans un autre objet tampon à l'aide de clCreateSubBuffer.
- La possibilité d'exécuter le noyau en mode synchrone et asynchrone.

- API pour lire, écrire et copier respectivement une région rectangulaire d'un objettampon :(clEnqueueReadBufferRect, clEnqueueWriteBufferRect and clEnqueue CopyBufferRect).

- Options qui contrôlent la version OpenCL C utilisée lors de la construction d'un exécutable de programme
- Des fonctionnalités d'OpenCL 1.0 sont obsolètes.
- Des nouvelles extensions sont ajoutées. La prise en charge des entiers 64 bits est facultative.

2.5.2.3 **Le standard OpenCL 1.2**

Juste un an après, le 15 novembre 2011, le groupe Khronos a ajouté une nouvelle version OpenCL 1.2 qui était la plus utilisée grâce aux améliorations en termes de fonctionnement et la programmation parallèle. On peut résumer les modifications dans les points suivants [38, 40, 41] :

- Les appareils personnalisés et les noyaux intégrés sont pris en charge.
- Partitionnement de périphérique qui permet de partitionner un périphérique en réduisant la latence pour les tâches critiques en réservant des zones mémoires.

Nouveaux types d'images : image 1D, image 1D à partir d'un objet tampon, tableaux d'images 1D et tableaux d'images 2D.

2.5.2.4 **Le standard OpenCL 2.0**

Le 18 novembre 2013 ; le groupe Khronos a publié la nouvelle version 2.0 publique des spécifications OpenCL ; les modifications ont concerné [38 , 42] :

- Mémoire virtuelle partagée.
- Files d'attente de périphérique utilisées pour mettre en file d'attente les noyaux sur le périphérique.
- Prise en charge des images pour l'image 2D à partir du tampon, des images de profondeur et des images RGB.
- Regroupement des noyaux dans une file d'attente de périphériques.
- Variables de portée de programme dans l'espace d'adressage global.
- Espace d'adressage générique.
- Prise en charge des images avec le qualificatif read_write.

- L'écriture d'images 3D est une caractéristique essentielle.
- La macro CL_VERSION_2_0.

2.5.2.5 **Le standard OpenCL 2.1**

Une autre version a été publiée par le groupe le 16 novembre 2015, et les apports majeurs par rapport à la version précédente sont [38, 43] :

- Le langage du noyau OpenCL C n'est pas mis à jour pour OpenCL 2.1.
- Le langage du noyau OpenCL 2.0 sera toujours utilisé par les runtimesOpenCL 2.1.
- Rajout des autres API pour donner d'autres fonctionnalités.
- La spécification SPIR-V a été ajoutée.

2.5.2.6 **Le standard OpenCL 2.2**

- Ajout de la troisième condition préalable (exécution de constructeurs non triviaux pour les variables globales de portée du programme).
- Le fonctionnement sur n'importe quel matériel compatible avec OpenCL 2.0 (seule la mise à jour du pilote est requise).
- Recommandation de l'utilisation intégrée du nouveau langage intermédiaire Khronos SPIR-V 1.1 [44;38] qui prend entièrement en charge les noyaux OpenCL C++ 14.

2.5.2.7 **Le standard OpenCL 3.0**

La spécification provisoire OpenCL 3.0 a été publiée le 27 avril 2020[45].OpenCL 3.0 réaligne la feuille de route OpenCL pour permettre aux fonctionnalités demandées par les développeurs d'être largement déployées par les fournisseurs de matériel, et il augmente considérablement la flexibilité du déploiement en permettant aux implémentations OpenCL conformes de se concentrer sur les fonctionnalités pertinentes pour leurs marchés cibles. OpenCL 3.0 intègre également des fonctionnalités de sous-groupe dans la spécification principale, est livré avec une nouvelle spécification de langage OpenCL C 3.0, utilise un nouveau format de spécification unifiée et introduit des extensions pour les copies de données asynchrones pour permettre une nouvelle classe de processeurs intégrés. Les spécifications provisoires d'OpenCL 3.0 permettent à la communauté des développeurs de fournir des commentaires avant la finalisation des spécifications et des tests de conformité.

OpenCL 2.X offre des fonctionnalités importantes, mais OpenCL 1.2 s'est avéré être la référence requise par tous les fournisseurs et marchés. OpenCL 3.0 intègre une option

étroitement organisée dans la spécification monolithique 2.2, augmentant la flexibilité de déploiement qui permettra à OpenCL de relever la barre des fonctionnalités disponibles de manière omniprésente dans les futures spécifications de base [41 ,45].

2.6 **Conclusion**

Comme présenté par ce présent chapitre, on a vu que, les machines parallèles sont caractérisées par une mémoire partagée ou distribuée. Ces deux catégories de mémoire sont à la base des bibliothèques qui réalisent le parallélisme, et dans le contexte bien particulier d'écriture de programmes parallèles, il existe de nombreux modèles de programmation. Parmi ces modèles de références, nous avons présenté MPI, un modèle de programmation destiné aux architectures distribuées, OpenMP pour le parallélisme multi-processeurs à mémoire partagée, Pthread qui permet une amélioration significative des performances informatiques en exploitant la puissance des processeurs CPU mais qui n'utilise pas l'unité de traitement graphique (GPU). La plateforme OpenCL se distingue par sa capacité à exploiter différents types de processeurs. Ce modèle de bas niveau permettent d'exploiter les capacités des machines parallèles. Contrairement au modèle de programmation parallèle traditionnel à mémoire partagée (OpenMP ,MPI,Pthread. . . etc) .Les interfaces traditionnelles pour la programmation graphique telles que DirectX et OpenGL, sont des *frameworks*dédiés aux calculs généralistes sur processeurs graphiques tandis que CUDA et OpenCL sont les principaux représentants de la programmation parallèle. OpenCL est un modèle de programmation particulièrement portable : il peut s'adapter autant aux GPUs qu'aux processeurs multicoeurs et manycore, le modèle de programmation présenté par OpenCL exige l'écriture de deux programmes séparé. En guide de conclusion nous précisons également que OpenCL incarne l'une des tentatives les plus abouties qui a pour ambition de proposer un modèle de programmation parallèle unifié pour l'ensemble des architectures et des dispositifs parallèles. Pour simplifier l'utilisation d'OpenCL, plusieurs concepteurs, conscients du potentiel réel pour la production d'applications parallèles hétérogènes, proposent des DESL (Domain Embedded SpecificLanguage) métaprogrammés afin de simplifier l'usage des APIs OpenCLet|ou de les optimiser. OpenCL est donc au cœur de notre étude, dans l'optique d'illustrer son fondement, son fonctionnement et son modèle de programmation.

Dans le chapitre 3, nous abordons une étude comparative des techniques d'optimisation de la programmation parallèle hétérogène CPU/GPU que nous appliquons au calcul l'algorithme du k-means. On présente une étude comparative de trois implémentations les plus performantes

de l'algorithme du K-means : l'implémentation séquentielle de l'algorithme du Lloyd-Forgy, lesimplémentations parallèles ciblant le CPU en utilisant OpenMP et Pthreadset enfin l'une des implémentations les plus complexes qui utilisent un Langage OpenCL.

2.7 **References:**

[1] David Kirk et Wen-mei Hwu. Programming Massively Parallel Processors, A hands-on Approach, second editon, USA: Morgan Kaufmann, Elsevier, Burlington USA, 2010.

[2] Flynn, M., *Some Computer Organizations and Their Effectiveness*, IEEE Trans. Comput., Vol. C-21, pp. 948, 1972.

[3] MICHAEL J. FLYNN AND KEVIN W. RUDD, *Parallel Architectures,* ACM Computing Surveys, Vol. 28, No. 1, March 1996

[4] Duncan, Ralph, *A Survey of Parallel Computer Architectures*, IEEE Computer. February 1990, pp. 5-16. http://www.spec.org

[5] wikipedia, «wikipedia,» wikipedia, 20 mai 2020. [En ligne]. Available: https://fr.wikipedia.org/wiki/Message_Passing_Interface.

[6] Michael McCool, Arch D. Robison and James Reinders. Structured Parallel Programming; Patterns for Efficient computation. Elsevier, Inc., Waltham, USA, 2012.

[7] Guy E. Blelloch. Vector Models for Data-Parallel Computing. The MIT Press, 1990.

[8] Peter Pacheco. An introduction to PARALLEL PROGRAMMING. Elsevier, 2011.

[9] M. Young, A. Tevanian, R. Rashid, D. Golub, and J. Eppinger. The duality of memory and communication in the implementation of a multiprocessor operating system. SIGOPS Oper. Syst. Rev., 21(5) :63–76, Novembre 1987

[10] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard Version 3.1," 2015. [Online]. Available: http://mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf

[11] The MPI Forum. The Message Passing Interface. url : http://www.mpi-forum. org/.

[12] The openmp api specification for parallel programming. Online available :http:// openmp.org/wp/.

[13] L. Dagum et R. Menon. Openmp : an industry standard api for shared-memory programming. IEEE Computational Science and Engineering, 5(1):46–55, Jan 1998. ISSN 1070-9924.

[14] N. Drosinos et N. Koziris. Performance comparison of pure mpi vs hybrid mpi-openmp parallelization models on smp clusters. 18th Int. Parallel and Distributed Symposium,, 2004.

[15] OpenMP Architecture Review Board. The OpenMP API specification for parallel programming. url : http://openmp.org/wp/.

[16] Open Group. POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995). url : http: //pubs.opengroup.org/onlinepubs/007904975/basedefs/pthread.h.html.

[17] David R. Butenhof, Programming with POSIX threads, USA: Addison Wesley, 1997.

[18] S. Cook, CUDA Programming, a developer's guide to parallel computing with

GPU's, U.S.A: MORGAN KAUFMANN, 2013.

[19] P. Patcheco, An Introduction to Parallel Programming-, San Francisco: Morgan KAUFMANN, 2011.

[20] J. Newburn Chris, So Byoungro, Liu Zhenying, McCool Michael, Ghuloum Anwar, Du Toit Stefanus, Wang Zhi Gang, Hui Du Zhao, Chen Yongjian, Wu Gansha, Guo Peng, Liu Zhanglin et Zhang Dan. Intel array building blocks :a retargetable, dynamic compiler and embedded language. Dans CGO 11 Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization, 2011.

[21] Intel. Intel threading building blocks (intel TBB) 4.3 update 3. Online available :https: //www.threadingbuildingblocks.org.

[22] S. SCHOLAR, «Deployment of parallel applications on a reconfigurable system on chip distributed architecture,» 2015. [En ligne]. Available: https://www.semanticscholar.org/paper/D%C3%A9ploiement-d'applications-parall%C3%A8les-sur-une-of-on-Ewo/c014f389e9a6fd857cb4c155eae66dfc741082cd. [Accès le 2020].

[23] Nir Shavit Maurice Herlihy. The Art of Multiprocessor ..Programming.. Elsevier, 2012.

[24] CUDA Nvidia. Programming guide, 2008.

[25] Aaftab Munshi. The opencl specification. In 2009 IEEE Hot Chips 21 Symposium (HCS), pages 1{314}. IEEE, 2009.

[26] Jianbin Fang, Ana Lucia Varbanescu, and Henk Sips. A comprehensive performance comparison of cuda and opencl. In 2011 International Conference on Parallel Processing, pages 216{225. IEEE, 2011.

[27] Cristobal A Navarro, Nancy Hitschfeld-Kahler, and Luis Mateu. A survey on parallel computing and its applications in data-parallel problems using gpu architectures. Communications in Computational Physics, 15(02):285{329, 2014.

.

[28] N. DEVELOPER, «NVIDIA DEVELOPER high performance computing,» NVIDIA DEVELOPER, 2020. [En ligne]. Available: https://developer.nvidia.com.

[29] John Cheng, Max Grossman and Ty McKercher, professional CUDA C Programming, ISBN: 978-1-118-73932-7 Canada: wiley Brand, 2014.

[30] a. developer, «apple developer ,metal,» Apple, 2020. [En ligne]. Available: https://developer.apple.com/metal/.

[31] «zdNet,» zdNet, 04 juin 2014. [En ligne]. Available: https://www.zdnet.fr/actualites/l-arrivee-de-metal-chez-apple-signe-t-il-la-mort-d-opengl-39801949.htm.

[32] S. I. UNIVERSITY, «l'API apple metal,» SUPINFO INTERNATIONAL UNIVERSITY, 23 Juillet 2018. [En ligne]. Available: https://www.supinfo.com/articles/single/7206-api-apple-metal.

[33] Benedict R. Gaster, Lee Howes, David R. Kaeli, Perhaad Mistry et Dana Schaa. Heterogeneous Computing with OpenCL, 2nd Edition Revised OpenCL 1.2 Edition, chaptitre Data Management, pages 147–159. Elsevier Inc, Waltham USA, 2011.

[34] Benedict Gaster, Lee Howes, David R Kaeli, Perhaad Mistry, and Dana Schaa. Heterogeneous Computing with OpenCL: Revised OpenCL 1. Newnes, 2012.

[35] Jonathan Tompson and Kristofer Schlachter. An introduction to the opencl programming model. Person Education, 2012.

[36] Introduction to opencl programming guide, 2010. Available at:http://developer.amd.com/tools-and-sdks/opencl-zone/opencl-resources/openclcourse-introduction-to-opencl-programming/ Accessed: 2016-03-05.

[37] k. g. w. group, «The OpenCL Specification version 2.1 Document Revision: 24,» 13 fevrier 2018. [En ligne]. Available: https://www.khronos.org/registry/OpenCL/specs/opencl-2.1.pdf.

[38] k. o. w. group, «the opencl specification,» 19 jul 2019. [En ligne]. Available: www.khronos.org/registry/OpenCL/specs/2.2/html/OpenCL_API.html#clEnqueueReadBufferRect.

[39] k. group, «guide opencl api 1.1 reference.,» [En ligne]. Available: https://www.khronos.org/files/opencl-1-1-quick-reference-card.pdf.

[40] khronos group, «carte, QUICK reference GUIDE, opencl API 1.2 REFERENCE,» [En ligne]. Available: https://www.khronos.org/files/opencl-1-2-quick-reference-card.pdf.

[41] N. Trevett, «kronos group connecting opencl to silicon,» kronos group, 2020. [En ligne]. Available: https://www.khronos.org/opencl/.

[42] k. group, «quick reference card opencl 2.0 reference,» [En ligne]. Available: https://www.khronos.org/files/opencl20-quick-reference-card.pdf.

[43] K. GROUP, «opencl2.1 REFERENCE GUIDE,» [En ligne]. Available: https://www.khronos.org/files/opencl21-reference-guide.pdf.

[44] khronos groupe, «OPENCL 2.2 reference guide,» [En ligne]. Available: https://www.khronos.org/files/opencl22-reference-guide.pdf.

[45] khronos group connecting softwae to silicon, «OpenCL 3.0 provisonal,» 26 april 2020. [En ligne]. Available: https://www.khronos.org/news/permalink/khronos-group-releases-opencl-3.0-provisional-specifications.

[46] LoiicFejoz. Developpement prouve de structures de donnees sans verrou. Thèse de doctorat, Henri Poincarie-Nancy-University, 2008.

[47] BangerRavishekhar et BhattacharyyaKoushik. OpenCL Programming By Example, pages 35–58. PACKT PUBLISHING, Birmingham UK, 2013.

[48] OpenCL Best Practices Guide , May 27, 2010.

[49] Tay, R.,. OpenCL Parallel Programming Development Cookbook. PacktPublishing.(2013).ISBN: 1849694524

[50] Gaster, B., Howes, L., Kaeli, D.R., Mistry, P., Schaa, D.: Heterogeneous Computing with OpenCL, 1st edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2011).

[51]James Reinders. Intel threading building blocks -outfitting {C++} for multi-coreprocessor parallelism. O'Reilly. 2007.ISBN: 978-0-596-51480-8.

[52] Michael Voss, Rafael Asenjo, James Reinders, (2019) Jumping Right In: "Hello, TBB!". In: Pro TBB. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4398-5_1.

[53] Microsoft Corp. *DirectX Graphics*. url : https://msdn.microsoft.com/enus/library/windows/desktop/hh309467.aspx.
[54] Khronos Group. *OpenGL : The Industry'sFoundation for High Performance Graphics*. url :https://www.khronos.org/opengl/.
[55] James Reinders Michael McCool, Arch D. Robison. Structured Parallel Programming , Patterns for Efficient computation. Elsevier, 2012.
[56] Microsoft. C++ amp (c++ accelerated massive parallelism). Online available :https:// msdn.microsoft.com/en-us/library/hh265137.aspx.
[57] Microsoft. C++ amp : Language and programming model version 1.2. Rapport technique, Microsoft Corporation, 2013.
[58] The OpenACC Consortium. The OpenACC Programming Interface. http://www.openacc-standard.org.
[59] Sandra Wienke, Paul Springer, Christian Terboven, and Dieter an Mey. OpenACC—first Experiences with Real-world Applications. In Euro-Par 2012 Parallel Processing, pages 859–870. Springer, 2012.

CHAPITRE 3 :
Etude comparative des implémentations parallèles CPU/GPU du K-Means

Sommaire

3.1 Introduction

Clustering est une méthode trés utile endata mining qui aide les analyseurs à comprendre le regroupement des attributs de données. Le clustering est la classification non supervisée de modèles tels que des observations, des éléments de données ou des vecteurs des caractéristiques dans des groupes appelés «clusters ».

Le clustering est largement utilisé dans de nombreux domaines tels que data mining, machine learning, la reconnaissance de formes, segmentation d'images, l'intelligence artificielle, l'analyse des données d'expression génétique et la bioinformatique.

K-means[1, 2] est une méthode non hiérarchique populaire pour le clustering de grands ensembles de données. Les exigences en matière du temps augmentent linéairement avec la taille de l'ensemble de données, ce qui le rend particulièrement adapté aux ensembles de données extrêmement volumineux tels que ceux que l'on trouve dans les bibliothèques numériques. La méthode a été développée par MacQueen [2] en 1967.

Dans de nombreux domaines d'étude, les scientifiques appliquent l'algorithme du k-means [1, 2] dans le but de détecter et de trouver des modèles dans de grands ensembles de données, il est également utilisé dans le data mining, traitement des images et machine learning. K-means a été classifié comme étant l'un des Dix meilleurs algorithmes de data mining [3]. Ces dernières années, les unités de traitement graphique (GPU) onts'est considérablement développée pour devenir une plateforme robuste, performante et prometteuse pour la parallélisation du K-Means. Comme nous le savons, les GPU sont des appareils exclusivement destinés à la manipulation des calculsgraphiques. En plus de cela, ils ont évolué en des processeurs parallèles supérieurs à multi-cœurs. Les performances de la puissance de calcul et la largeur de bande de la mémoire des GPU ont conduit à l'utilisation de calculs généraux [4].

Par conséquence, K-means est une étude de cas qui continue à attirer l'intérêt des chercheurs et des spécialistes pour l'optimisation et le calcul parallèle.

Dans ce chapitre, on présente une étude comparative de trois implémentations les plus performantes de l'algorithme du K-means : l'implementation séquentielle de l'algorithme du

Lloyd-Forgy, lesimplémentations parallèles ciblant le CPU en utilisant OpenMP / Pthreadset enfin l'implementation complexe qui utilise un Langage OpenCL.

Le reste du chapitre est organisé en différentes sections.La section 2 présente les résultats obtenus précédemment dans l'accélération de l'algorithme du K-Means. La section 3 est une brève introduction de l'algorithme du K-Means ainsi qu'une description détaillée de l'algorithme parallèledu K-Means avec OpenCL et OpenMP/ Pthreads. La section 4 décrit les résultats exprimentaux. Enfin, la section 5 conclut l'étude et présente nos intentions de recherche à venir.

3.2 **Travauxconnexes**

La parallélisation de l'algorithme du k-means s'inscrit dans ce contexte. L'objectif est de réduire le temps de calcul lorsque cet algorithme est appliqué à de petits et grands volumes de données. À ce jour, plusieurs travaux adoptès à la parallélisation de l'algorithme du k-means, ont été signalés. On utilisant le framework Message Passing Interface (MPI) [5] et MapReduce [6] sont parmis les modèles réussis de parallélisation de l'algorithme du K-Means.

Dans les deux approches [7, 8], les auteurs proposent un algorithme parallèle du K-means basé sur MPI, et les bénéfices apparents du parallélisation sont clairement remarquables.

L'architecturehétérogène a commencé à jouer un rôle important dans l'algorithme du K-means, que ce soit sur le FPGA [9], [10], le GPU [11], [12], [13] ou d'autres plateformes. Les résultats ont permis de conclure qu'une forte accélération a été obtenue.

L'approche de *Fang et al* [14] proposent une implémentation de l'algorithme du K-means basé sur le GPU. Ils affirment que le GPU utilise la mémoire constante pour stocker les K-centroïdes et copier les données dans la mémoire de texture, de sorte que chaque point de données est affecté à un thread et utilise le cache. Cependant, nous pouvons améliorer cette implémentation par d'autres accès à la mémoire tels que la mémoire partagée.

Dans l'approche prèsentèe dans [15], les auteurs proposent une nouvelle version de l'algorithme du K-means dépendant de l'architecture à instructions multiples et données multiples (MIMD) via plusieurs GPUs ou le streaming multiprocesseur (SM). Ils ont calculé la distance et l'étape de recalculer les K- Centroïdes en parallèle dans le GPU. Dans ce cas, le CPU n'effectuera que les étapes d'initialisation du k-clusters et le test de convergence. Cette implementation repose sur l'utilisation des registres du GPU, la mémoire de texture, la mémoire constante et la mémoire partagée. En outre, cette implementation nécessite plusieurs GPUs.

L'utilisation de GPU pour paralléliser les K-means a été proposée dans [24] ou on utilise un nouvel algorithme de réduction irrégulière dans lequel à la fois le Map et Reduce peuvent être calculées sur le GPU avec un minimum de transfert de données entre le CPU et le GPU. Et utilisé la bibliothèque OpenCL pour atteindre une accélération de 3,2 pour 10 centroïdes et 1,5 pour 100 et 400 centroïdes.

L'approche de Zhao et al [22] propose un algorithme parallèle rapide du K-means basé sur MapReduce, qui peut être mis à l'échelle de manière efficace pour traiter de grands ensembles de données.

Dans L'approche de [21] les auteurs ont implémenté l'algorithme du K-means sur Stratix VA7 FPGA , ils utilisent deux kernels le premier "assignement kernel" chargé pour calculer la distance et le deuxième "reduce kernel" pour la mis à jour des clusters ,plus la taille du cluster augmente alors le kernel fonction moins (mal) ,les performances de FPGA sont comparées a celle de l'implémentation GPU , le FPGA est plus performant que le GPU pour les tailles de données moyennes et petit mais moins lente qu'un GPU pour les taille de donnèes plus grandes .en plus l'efficacité énergétique FPGA a plus de 29 fois qu'un CPU et meilleure qu'un GPU .

Une implémentation FPGA [23] de l'algorithme k-means utilisant MapReduce a été présentée en 2014. Les calculs de l'algorithme K-means sont divisés en des fonctions Map et Reduce et implémentés dans des FPGA séparés. Deux FPGA sont utilisés pour Mapper et un FPGA pour Reduce, les résultats montrent que l'implémentation multi-FPGA peut surpasser le logiciel de base mise en œuvre dans tous les cas de test, en offrant une accélération de 15,5 à 20,6.

Le travail de Li et al [25] vise à paralléliser les K-means, qui utilisent deux stratégies différentes pour les ensembles de données à petite dimension et les ensembles de données à grande dimension. Pour les ensembles de données à petite dimension l'algorithme exploite les registres du GPU pour réduire de manière significative la latence d'accès aux données. Pour les ensembles de données à grande dimension, l'algorithme simule une multiplication matricielle et exploite la mémoire partagée du GPU pour obtenir un rapport d'accès calcul/mémoire élevé.

Des chercheurs d'AMD [26] ont présenté une nouvelle implémentation pour GPU OpenCL du K-means. Ils ont utilisé des réductions irrégulières et effectuaient le calcul complètement sur le GPU. Ils ont obtenu une accélération de plus de 35 fois par rapport à un CPU à quatre cœurs pour les données volumineuses et ils ont affirmé être 3,2 fois plus rapides, et 1,5 fois aussi rapides que les implémentations hybrides CPU-GPU pour des tailles de clusters de 10, 100 et 400, respectivement, sur ATIHD5870. Cependant, lorsque la taille du cluster augmente, la vitesse de performance diminuepour cette implémentation.

Des chercheurs de l'Université Indiana ont présenté une implémentation OpenCL des d'algorithmes statistiques itératifs, y compris k-means, qui ciblant les GPU NVIDIA Tesla. [27]. La mémoire locale et l'optimisation des modèles d'accès ont été appliquées pour augmenter les performances du noyau à un peu plus de 100 GFLOPS/s avec des données bidimensionnelles, 300 centroïdes et plus d'un million de points de données. Cette implémentation utilise un GPU pour calculer l'affectation des clusters et le processeur pour mettre à jour les centroïdes. Le GPU utilisé dans le test de performance est un Tesla C1060 qui assure théoriquement 933 GFLOPs/s en simple precision est similaire à la GTX280.

Comme pour les plates-formes de parallélisation bien connues, CUDA s'avère être fortement contraint uniquement par la plate-forme NVIDIA. D'autre part, le langage Open Computing Language (OpenCL) [16], qui est un standard industriel multi-plateforme, nous a offert un choix plus efficace et plus compatible.

3.3 Les implémentations conventionnelles du K-means

3.3.1 L'algorithme séquentiel du K-means

L'algorithme du K-Means [1], [2] est défini comme un algorithme du clustering non supervisé. Il classifie N points de données d'entrée de D attributs en K clusters en fonction de leur distance inhérente les uns par rapport aux autres. Le nombre K est fixé à l'avance. Chaque ensemble est représenté par un moyen (ou centre) calculé à partir de tous les autres points de cet ensemble de données. Cependant, les principaux inconvénients de K-means se déroulent progressivement que le résultat repose sur le centres initiaux de clusters. Alors la meilleure initialisation des centresva converger plus vite et avoir moins d'itérations. C'est évident que les résultats finaux seront affectés par ces inconvénients sérieux.Les étapes de l'implementation séquentielle de l'algorithme du K-means sont présentées ci-dessous :

Algorithme 3.1 : Algorithme K-Means séquentiel

Entrée : K (nombre de clusters), ensemble de N points de D dimension,

Sortie : partition de *N points en K clusters.*

1 : Placer les centroïdes c_1, c_2, ..., c_K à des endroits aléatoires

2: Iterate until convergence condition is met

3: Pour chaque point x_i, i=1..N :

4: Pour chaque cluster c_j, j=1.. K :

5: Calculer la distance à c_j, compte tenu de toutes les dimensions D

6 : Attribuer chaque point xi au cluster j le plus proche

7 : Pour chaque cluster c_j, j=1... K :

8 : Recalculer les Centroïdesc_j =moyenne de tous les points dont il fait partie de J

O(#itérations x N x D x K)

En 2001, *Estlick et al.* ont évalué de nombreux types de mesures de distance et sont arrivés à la conclusion suivante : bien que la distance euclidienne soit la plus précise pour le KMeans, la distance de Manhattan est mieux adaptée à la conception du matériel en raison de la réduction des congestions et de la vitesse plus élevée.

Tous les types de distance peuvent être acceptés, parmi lesquels les distances de Manhattan et Euclidienne sont souvent considérées .

L'algorithme du K-Means tend à converger vers le minimum local qui dépend de la façon dont les centroïdes ont été sélectionnés initialement. Nous concluons que la sélection des centroïdes initiaux est très importante et doit être effectuée avec précision dès le départ.

Pour chaque itération de l'implémentation du k-means, la complexité de calcul (dans le pire des cas) est de O (D*N*K +N*K + N*D), où K est le nombre de cluster et N est l'ensemble des points avec D dimensions. La partie la plus intensive en calcul de cette implémentation est l'étape de calcul de la distance qui nécessite une soustraction, une addition et une multiplication pour calculer la somme partielle de la distance pour chaque point de données. D'une manière générale, le nombre d'opérations est approximativement égal au nombre d'itérations*D*N*K*3.

3.3.2 **Les implementations parallèles de l'algorithme du K-means avec OpenMP et pthreads**

3.3.2.1 **Parallélisation basée sur les threads POSIX en C:**

Pour la parallélisation de l'algorithme du k-means basée sur les pthreads, une approche de parallélisme des données a été adoptée. Les N points ont été répartis de manière égale entre les nombre de threads (num_threads) générés à l'aide de ***pthread_create*** (en cas de répartition inégale entre les threads, les points restants sont alloués au dernier thread).

Points essentiels de la parallélisation de l'algorithme :

- **Initialisation** : Les K premiers points de données sont choisis aléatoires comme centroïdes initiaux
- **Parallélisme des données :** Chaque thread assigné N/num_threads nombre de points
- **Fonction de thread :** Chaque thread exécute une boucle, dans chaque itération de laquelle il calcule le centroïde de cluster le plus proche pour chaque point qui lui est attribué, puis attribue le point à ce cluster.
- **Exclusion mutuelle et la section critique** : La première stratégie de synchronisation des threads qui garantit que les mises à jour du tableau de centroïdes du cluster partagé sont effectuées par chaque thread seulement après l'acquisition d'un mutex ou d'un spinlock (si la compilation est faite avec le flag DUSE_SPINLOCK). Ceci est essentiel pour éviter la course aux données due aux différents threads en essayant de modifier le tableau partagé.
- **Barrières :** Après chaque itération, des barrières sont utilisées pour la synchronisation des threads afin que tous les threads affichent la même valeur actualisée du tableau

des centroïdes. Des barrières sont utilisées pour la synchronisation de la mise à jour des centroïdes.

3.3.2.2 **-La parallélisation basée sur OpenMP en C**

OpenMP est une interface de programmation d'application standard en C, C++ et FORTRAN utilisée pour la programmation parallèle à mémoire partagée. Elle fournit un modèle de programmation parallèle portable sur les architectures à mémoire partagée. Les compilateurs de nombreux fournisseurs prennent en charge l'API OpenMP.

Un algorithme K-Means de partitionnement parallèle en utilisant OpenMP est présenté pour effectuer une analyse de cluster dans des systèmes multi-core pour de très grands ensembles de données. Les données sont divisées de manière égale en L partitions. Pour chaque partition, initialiser le processus OpenMP pour effectuer le traitement parallèle dans les systèmes multi-cœurs. Pour chaque partition, les valeurs moyennes sont calculées pour former K clusters locaux jusqu'à ce que la convergence soit atteinte. Les résultats obtenus à chaque K clusters locaux formés à P processus sont synchronisés pour appliquer un mécanisme de réduction globale pour construire un K clusters global. Cette procédure est continuée jusqu'à ce que la convergence soit atteinte.” Voir l’algorithme 3.2”.

Le benchmark MineBench [18] comprenait une implémentation OpenMP/OpenMPI multithreade CPU construisant un algorithme parallèle du k-means. La version OpenMP du code de référence de l'algorithme du k-means est utilisée dans ce chapitre afin de comparer les performances avec l’implementation sequentielle CPU.

La stratégie de parallélisme des données est identique à celle adoptée pour la parallélisation basée sur OpenMP. La synchronisation des threads est effectuée à l'aide d'une construction critique pragma omp pour l'exclusion mutuelle et d'une barrière pragma omp pour la synchronisation.

Les étapes de l’implementation de l’algorithme de l’OpenMP K-means sont présentées par la figure 3.1.

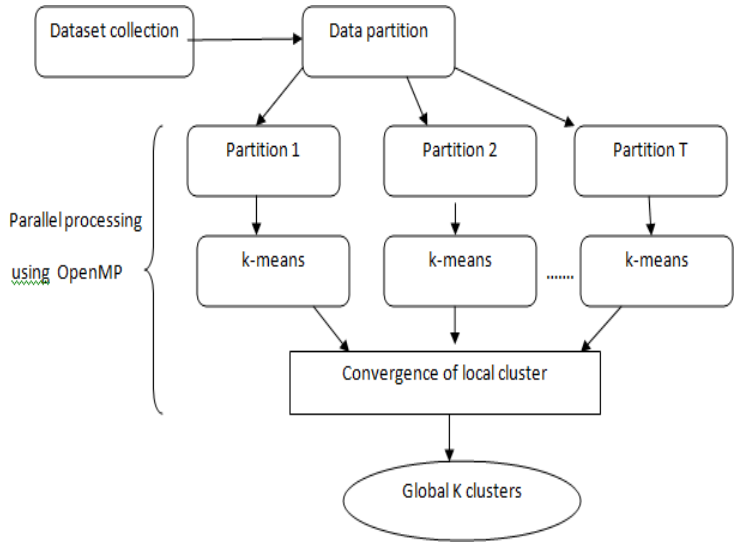


Figure 3-1 : Conception structurelle d'un algorithme parallèle du k-means en utilisant OpenMP.

3.3.3 L'implementation parallèle de l'algorihme du K-means avec OpenCL

Cette section décrit l'algorithme parallèle du K-means basé sur le GPU. L'algorithme K-Means est une méthode itérative. Ainsi, le calcul de la distance entre chaque point et les différents clusters montre beaucoup de parallélisme à l'intérieur de chaque itération. Les étapes 2 à 6 représentent la partie qui nécessite une accélération pour explorer le parallélisme.

Les étapes de l'implementation paralléle de l'algorithme du K-means en utilisant OpenCL sont présentées ci-dessous (voire figure 3.2 et l'algorithme3.3) :

Algorithme3.2: l'algorithme k-meansparallel en utilisant openMP

Entrée : D={d1,d2,d3,d4,...dn}K= Clusters initiaux

Sortie : K Clusters

Procédure :

1. Initialiser les processus OpenMP comme P ={P0, P1,P2,.... Pn}
2. Partitionner l'ensemble de données trié en k parties et assigner chaque partie de l'ensemble de données à chaque processus
3. Pour chaque partition, calculer les valeurs moyennes et former les k clusters locaux.
4. Continuez le processus pour former les K clusters locaux finaux jusqu'à ce que la convergence soit atteinte.
5. Maintenant, synchroniser les résultats du processus individuel.
6. Puis obtenir les K clusters globaux à partir de tous les K clusters locaux.
7. Continuez la procédure jusqu'à ce que la convergence soit atteinte.

Étape 1: L'étape d'initialisation des centres de clusters est essentielle, car les résultats peuvent varier en fonction du point de départ. La méthode la plus répandue consiste à choisir au hasard les K vecteurs de l'ensemble de données comme centres de clusters initiaux.

Donc sélectionner au hasard des vecteurs comme centres de clusters initiaux pour notre conception. Ensuite, nous procédons de charger et lire les données. Le nombre d'itérations défini auparavant est utilisé comme critère de convergence.

Étape 2 : Dans cette étape, on doit copier les données et les k-clusters choisis de la mémoire de CPU vers le GPU .

Étape 3 : Les N-threads correspondent à N-données. Respectivement, les N-threads calculent la distance entre les N-données et les K-centres de clusters. Afin d'économiser la bande passante de la mémoire globale, chaque cluster est chargé une fois par son groupe de travail respectif, dans la mémoire locale puis partagé entre tous les threads du groupe de travail.

Étape 4 : En utilisant la distance euclidienne, les données sont classées en fonction du centre de cluster le plus proche et ces résultats partiels sont copiés dans la mémoire du GPU.

Étape 5 : Recalculer les nouveaux centres de cluster. Comptez le nombre d'itérations défini précédemment, une fois le nombre d'itérations maximum atteint, passer à l'étape suivante, sinon revenez à l'étape 3.

Étape 6 : Les résultats du clustering sont complets.

Algorithme 3.3 : Pseudocode de l'algorithme parallèle du K-Means en utilisant OpenCL.

```
Function kmeans –kernel (Workgroup size= Nclusters* Ndim ) is
    Input :  Points,clusters ;
    Output :  Menbership ;
    gid ← get_global_id(0);
    lid ← get_local_id(0);
    cluster_local[Nclusters* Ndim] ;
    cluster_local[lid] ←  clusters[lid] ;
     index ← 0 ;
     min_dist ← 8  ;
    for K € [0, Nclusters] do
        dist ← 0 ;
        for D € [0, Ndim] do
            dist +=( points[D * Npoints + gid] - cluster_local [K* Ndim +D] )*
                ( points[D * Npoints + gid] - cluster_local [K* Ndim  +D] );
        end
          if (dist < min_dist) then
                                min_dist ← dist ;
                                index ← K ;
    end
    membership[gid] ← index;
end
```

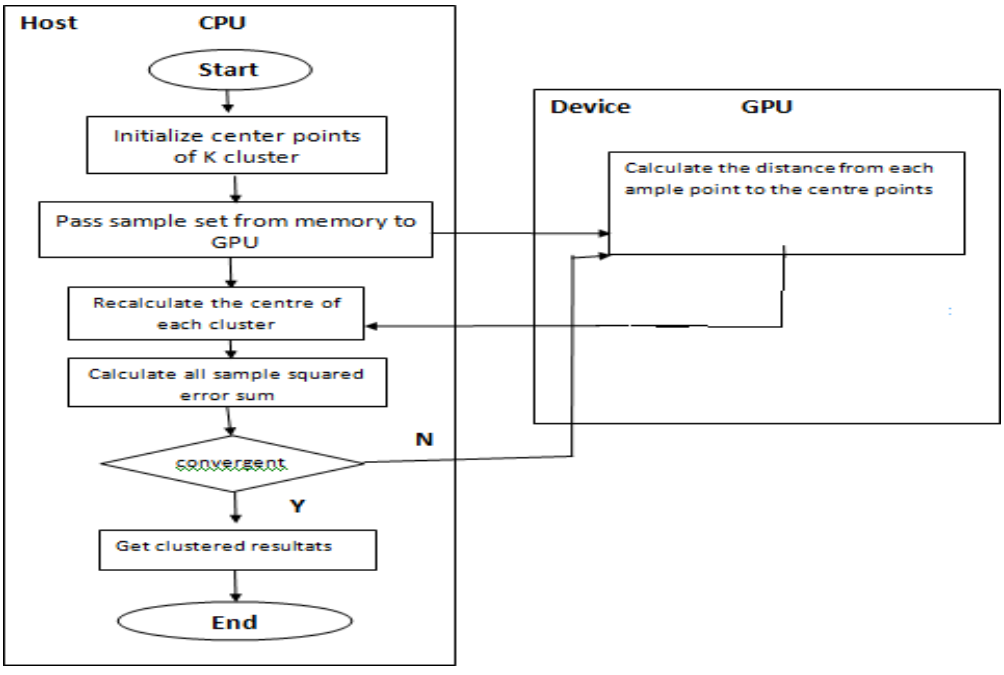


Figure 3-2: Les Étapes de l'implémentation parallele de l'algorithme du K-means en utilisant OpenCL

3.4 Les résultatsexpérimentaux

Les résultats de l'expérimentation ont été obtenus à partir d'ensembles de données préalablement classifiés. En outre, les points de données (N) et les attributs (dimentions (D)) varient. Les ensembles de données sont générés en faisant varier uniquement le nombre total de points et les paramètres des dimensions [19].

Les datasets de 8192, 65536 et 1000000 points de données sont calculés avec 2, 4,16 et 128 dimensions. En outre, ces ensembles de données sont calculés pour différents nombres de clusters(K) (2, 16, 128, 512 et 1024).

Les k-centroïdes initiaux sont choisis en utilisant la sélection aléatoire des vecteurs de l'ensemble de données.

La convergence est définie comme une situation idéale où il n'y a pas de réattribution des points de données après un nombre fini d'itérations (iter). Nous avons choisi comme critère de terminaison le nombre d'itérations prédéfini préalablement

Dans ce chapitre, le nombre d'itérations défini au préalable est utilisé comme critère de convergence. Cela est dû aux raisons suivantes. Premièrement, notre objectif principal est d'explorer l'accélération de l'implémentation sur GPU par rapport à l'implémentation sequentielle. Lorsque le nombre d'itérations est fixe, il est beaucoup plus facile d'identifier le facteur d'accélération. Ensuite, il est plus facile d'implémenter cette terminaison sur le matériel et de réduire l'utilisation de ressources matérielles supplémentaires. Enfin, c'est la meilleure solution pour le débogage et le tests si nous avons besoin de mettre fin à cet algorithme..

Nous avons comparé les performances de cette implémentation en GPU avec une implémentation du k-means en CPU. Les mêmes ensembles de données ont été utilisés pour les exécutions sur GPU et CPU.

Notre objectif est d'exploiter notre environnement hétérogène de calcul, qui intègre un CPU Intel i7-3.40GHZ, un NVIDIA Geforce GTX 960, et qui est programmé avec OpenCL Version 1.2. pour exécuter du K-means.

Pour être plus précis, nous avons utilisé le modèle de programmation OpenMP et nous l'avons compilé avec la version du compilateur Intel C/C++ (ICC) afin de mettre en parallèle l'algorithme.

Cette section présente et analyse les résultats de ce travail. Nous avons comparé le temps d'exécution de la version de l'algorithme parallèle du GPU utilisant OpenCL sous la plate-forme Ubuntu 16.04 avec la version séquentielle de l'algorithme du CPU et les implémentations parallèles ciblant le CPU utilisant OpenMP et Pthreads. Les résultats sont présentés dans les figures 3.3, 3.4 ,3.5 et 3.6.

En comparant l'algorithme k-means basé sur OpenMP et Pthreads [20] voir la figure3.6, les valeurs de l'accélération relative sont un peu similaires. Habituellement, l'implémentation basée sur OpenMP est légèrement plus performante que celle basée sur Pthreads pour un nombre de threads plus élevé. En ce qui concerne la parallélisation de l'algorithme OpenMP basé sur k-means, la vitesse augmente avec l'augmentation du nombre de threads mais sature à 8 threads, et donc la meilleure performance est à num_threads = 8.

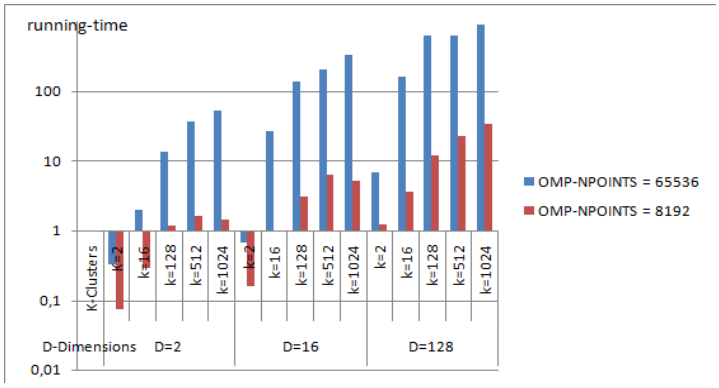


Figure 3-3: L'algorithme parallèle du K-means utilisant OpenMP à 8 threads en variant les data-sets , le nombre de clusters , le nombre de dimensions et le nombre d'itérations fixé à 20.

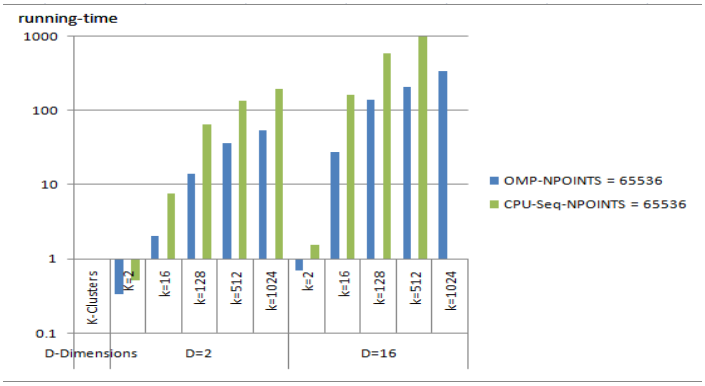


Figure 3-4: Comparaison entre l'implémentations parallèle du K-means en utilisant OpenMP(8 threads) et l'algorithme séquentiel, à un nombre d'itérations fixé à 20.

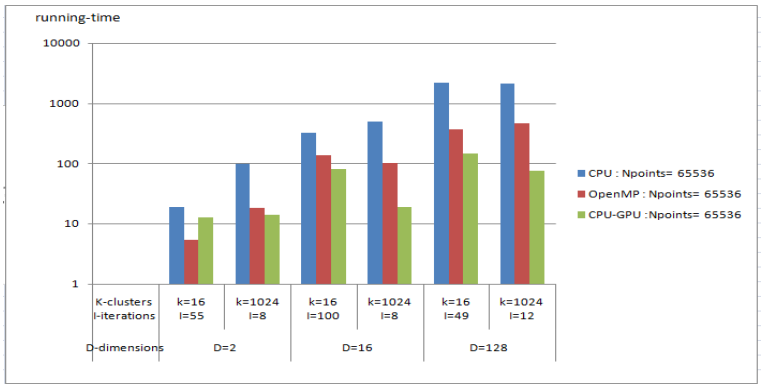


Figure 3-5: Data-set fixé à N = 65536, le nombre d'itérations fixé à 20 : Comparaison entre différentes implémentations parallèles avec la version séquentiel.

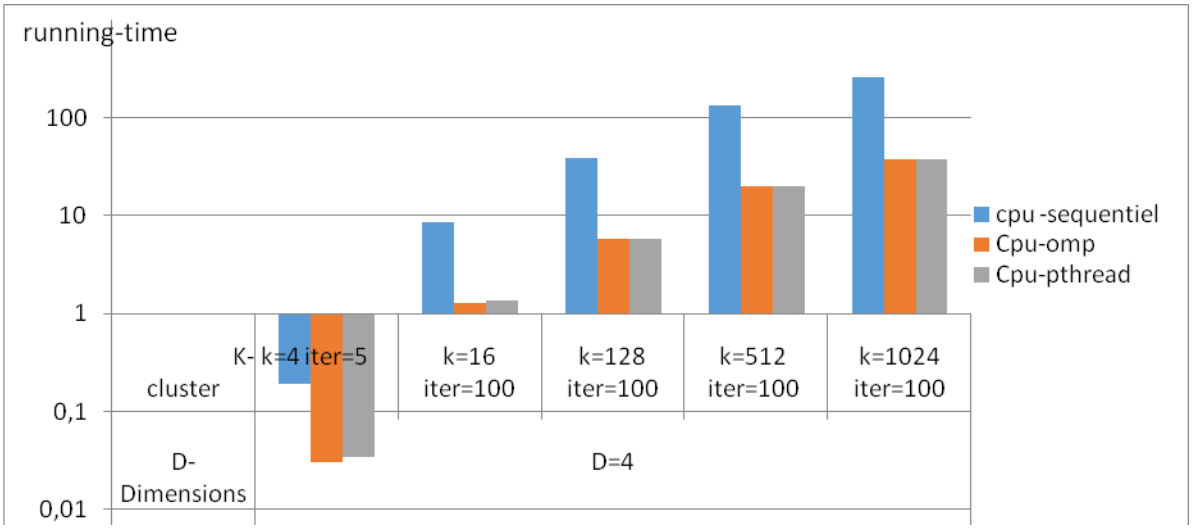


Figure 3-6: Data-set fixé à N = 1000000 et D=4: Comparaison entre différentes approaches parallèles avec la version séquentielle.

Deux conclusions peuvent être tirées à partir des figures précédentes :

(1) Lorsque le nombre de centres de cluster est inférieur ou égal à 16 et que la taille des données est petite, la méthode parallèle permet d'augmenter la transmission des données. Cela est dû au fait que les données sont copiées de la mémoire vers le GPU ; les résultats de la mise en clusters partielle sont copiés du GPU vers la mémoire. Par conséquent, le temps de calcul augmente.

(2) Avec l'utilisation d'une plus grande taille de données et une augmentation du nombre de centres de clusters, l'avantage de la méthode parallèle se fait sentir. Lorsque K = 1024, le ratio de la vitesse est de **3,48**.

3.5 Conclusion

En conclusion, l'algorithme parallèle du k-means proposé dans ce chapitre sur le GPU permet d'obtenir un parallelism tres important. Lorsque K = 1024 et la dimension D = 128, le ratio

d'accélération est de **3,48**, et le meilleur choix est donc atteint lorsqu'il s'agit de traiter de grands ensembles de données (figure 3.5). Par contre, le traitement de petits ensembles de données sur le CPU en utilisant d'OpenMP conduit au meilleur choix,lorsque K = 16 et dimension D = 2, le ratio d'accélération est de 2, car en raison du temps de transfer d'entrée/sortie est important, (figure 3.2). Le développement d'une version FPGA intégrée serait intéressant afin d'analyser plus en détail le compromis entre le temps d'exécution total et la consommation d'énergie.

Dans le chapitre 4, nous présentons une étude complète sur leparallélismede l'algorithme K-means++. Nous proposons une nouvelle implémentation parallèle de l'algorithme K-means++ en utilisant l'unité de traitement graphique (GPU). La plateforme Open Computing Language (OpenCL) est utiliséecomme environnement de programmation avec l'utilisation de la technologie Streaming SIMD Extension (SSE).

3.6 **Réferences**

[1] Lloyd, S. Least Squares Quantization in PCM. IEEE , 129–137(1982).

[2] MacQueen J et al Somemethods for classification and analysis of multivariate observations. In: Proc. Fifth Berkeley Symp. On Math. Statist. and Prob., vol 1, pp281–297(1967).

[3] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14.1–37(2008).

[4] S.A.A. Shalom, M. Dash, and M. Tue, “Efficient K-MeansClusteringUsingAccelerated Graphics Processors,” Proc. 10 th Int’lConf. Data Warehousing and KnowledgeDiscovery I. Song,eds.,pp. 166 175, (2008).

[5] Dhillon, I.S., Modha, D.S.: A data-clusteringalgorithm on distributed memory multiprocessors.LNCS (LNAI), vol. 1759, pp. 245–260. Springer(2000).

[6] Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In:Proceedings of the 6th Conference on Symposium on Operating Systems Design &Implementation, 6, p. 10. USA(2004).

[7] J. Zhang, G. Q. Wu, X. G. Hu, S. Y. Li, and S. L. Hao, "A parallelKmeanscluteringalgorithmwith MPI," IEEE Computer Society Washington, DC, USA, (2011).

[8] M. N. Joshi, "Parallel K-meansalgorithm on distributed memory multiprocessors,"

Computer,(2003).

[9] Sharafeddin M, Saghir MA, Akkary H, Artail H, Hajj H On the effectiveness of acceleratingMapReducefunctionsusing the XilinxVivadoHLS tool. Int J High PerformSystArchit 6(1):1–12(2016).

[10] Sharafeddin M, Partamian H, Awad M, Saghir M.A, Akkary H, Artail H, Hajj H, Baydoun M Towardsdistributedacceleration of image processing applications using reconfigurable active SSD clusters: a case study of seismic data analysis. Int J High Perform Comput Networking (2016).

[11] L. Mussi, F. Daolio and S. Cagnoni, "Evaluation of parallelparticleswarmoptimizationalgorithmswithin the CUDA (TM) architecture," Information Sciences, vol. 181(20), pp. 4642-4657,(2011).

[12] L.P. Veronese, R.A. Krohling, "Swarm's flight: Accelerating the particlesusing C-CUDA," IEEE, pp. 3264-3270,(2009).

[13] L. Mussi, S. Cagnoni, "ParticleSwarmOptimizationwithin the CUDA Architecture" GECCO conference,(2008).

[14] W. Fang, K. K. Lau, M. Lu, X. Xiao, C. K. Lam, P. Y. Yang, B. He, Q. Luo, and K. Yang, “Parallel Data Mining on Graphics Processors”, Technical Report HKUSTCS08,2008.

[15] H. Fakhi, O. Bouattane, M.Youssfi and O.Hassan. « New optimized GPU version of the k-meansalgorithm for large-sized image segmentation »IEEE (2017).

[16] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallelprogramming standard for heterogeneouscomputingsystems," Computing in Science and Engineering, vol. 12, pp. 66-73,(2010).

[17] M. Estlick, M. Leeser, J. Theiler and J. J. Szymanski, "Algorithmic Transformations in the Implementation of K-meansClustering on Reconfigurable Hardware," in ACM/SIGDA 9th International Symposium on FPGA '01,(2001).

[18] Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary. MineBench: A benchmark suite for data miningworkloads. *IEEE International Symposium onWorkloadCharacterization*, 182–188, 25–27.(2006).

[19] http://www.cs.virginia.edu/~kw5na/lava/Rodinia/Packages/Current/rodinia_3.0/data/kmeans/.

[20] https://github.com/arneish/parallel-k-means.

[21] Tang, Q. Y., & Khalid, M. A. . Accelerationof K-MeansAlgorithmUsingAltera SDK forOpenCL. ACM TransactionsonReconfigurableTechnologyandSystems(TRETS), 10(1), 6. 2016.

[22] W. Zhao, H. Ma, Q. He, "Parallel K-Means Clustering Based on MapReduce in Clo ud Computing," vol. 5931, pp. 674-679, 2009.

[23] Choi and H. So. Map-reduceprocessing of k-meansalgorithmwith FPGA-accelerated computer cluster. In IEEE 25th International Conference on Application-SpecificSystems, Architectures and Processors.2014.

[24] Dhanasekaran and N. Rubin.. A new method for GPU basedirregularreductions and its application to k-meansclustering. In Proceedings of the 4th Workshop on General PurposeProcessing on Graphics ProcessingUnits (GPGPU-4'11). ACM, New York, NY, 2011.

[25] You Li, Kaiyong Zhao, Xiaowen Chu, and Jiming Liu. Speeding up k-meansalgorithm by GPUs. Journal of Computer and System Sciences 216–229© 2012 Elsevier.

[26] B. Dhanasekaran and N. Rubin. 2011. A new method for GPU basedirregularreductions and its application to k-meansclustering. In Proceedings of the 4th Workshop on General PurposeProcessing on Graphics ProcessingUnits (GPGPU-4'11). ACM, New York, NY, 2011.

[27] Gunarathne, B. Salpitikorala, G. Fox, and A. Chauhan. 2011. OptimizingOpenCLkernels for iterativestatistical applications on GPUs. In Proceedings of the 2nd International Workshop on GPUs and ScientificApplications (GPUScA'11).

CHAPITRE 4 :

Parallélisassion de l'algorithme de clustering du k-means++

Sommaire

4.1 **Introduction**

Le clustering est l'une des tâches fondamentales et descriptives de data mining. Le clustering est une méthode d'apprentissage non supervisée qui consiste à regrouper un ensemble d'objets dans une même classe [1]. L'analyse des clusters dépend de l'attribution d'un ensemble d'objets à des sous-ensembles (appelés clusters) de sorte que les objets d'un même cluster soient identiques selon des critères prédéfinis [2]. Diverses applications se trouvent dans [3-4].

L'algorithme du clustering k-means développé par McQueen en 1967 [5], l'un des algorithmes de clustering non supervisé les plus simples, attribue chaque point d'un cluster au centre (centroïde) le plus proche. En général, cet algorithme comprend trois étapes : l'initialisation, le traitement et laconvergence.

Le centre correspond à la moyenne de tous les objets appartenant au cluster ; leurs coordonnées correspondent à la moyenne de chaque caractéristique séparément de tous les objets du cluster, où chaque cluster est représenté par son centroïde. Cependant, il est important de noter que les auteurs de [6] ont déterminé que l'implémentation du K-means convergera vers un optimum local ; la sélection des premiers centres de cluster a été cruciale lors de l'implémentation de l'algorithme du K-means. Le défi consiste à assurer une meilleure

initialisation des centroïdes. Il existe plusieurs approches pour atteindre cet objectif. Dans [7], ils ont proposé d'utiliser une technique intelligente pour déterminer les centroïdes initiaux du k-means, ce qui a conduit à un algorithme combiné appelé k-means++. k-means ++ s'est rapidement prouvé être l'un des algorithmes le plus populaires, avec des implémentations dans des domaines differents tels que le clustering d'informations géographiques [8], les microblogs [9], les réseaux sociaux [10] et la compression d'images [11].

k-means++ choisit arbitrairement les centroïdes initiaux en se basant sur une approche de probabilité proportionnelle en calculant la distance de chaque point de données par rapport aux centroïdes existants. Pour sélectionner les k centroïdes initiaux, K-means++ a besoin de k-itérations, et pour cette raison, cette méthode pourrait ne pas être efficace pour l'ensemble de données comportant un grand nombre de clusters.

Bahmani et al. [12] présentent une autre méthode d'initialisation de L'algorithme du K-means, néanmoins, ils ne fournissent pas de comparaisons du temps entre leur implémentation et K-means et même K-means++. Le problème le plus notable de cet algorithme est lorsque l'exécution se fait en mode séquentiel, car cela réduit la rapidité du clustering. Dans notre travail, nous choisissons d'exploiter les unités de traitement graphique (GPU) car elles garantissent des performances plus élevées. En plus, les GPU vont être progressivement utilisées pour des implémentations en temps réel de l'algorithme du K-means [13-16].

Dans ce chapitre, nous proposons une nouvelle implémentation parallèle de l'algorithme du K-means ++ en utilisant le GPU. La plate-forme Open Computing Language (OpenCL) [17,18] est utilisée comme environnement de programmation en combinaison avec l'utilisation de la technologie Streaming SIMD Extension (SSE) [19], qui met l'accent sur les optimisations directement liées à cette architecture pour exploiter au maximum les capacités informatiques du calcul disponibles. K means++ est un algorithme itératif dans lequel chaque itération comprend deux phases : la phase d'attribution des données et la phase mise à jour des k-centroïdes. Pour accélérer les parties du k-means++ qui nécessitent beaucoup du calculs, l'étape d'initialisation pour sélectionner les k-centroïdes initiaux est exécutée en parallèle à l'aide des instructions SSE tandis que l'étape d'attribution des données est effectuée en parallèle au niveau du GPU. Seulement les étapes de recalculer des K-centrroïdes et de test de convergence sont effectuées par le CPU.

Notre contribution se présente donc comme suit :

1. Nous présentons un nouvel algorithme qui concerne l'étape d'initialisation pour sélectionner les k-centroïdes initiaux en parallèle en utilisant la technologie SSE.
2. Nous présentons un nouvel algorithme d'attribution des données pour mettre à jours les centroides en parallèle à l'aide du GPU, la plate-forme OpenCL est utilisée, tout en employant la mémoire locale et la mémoire privée du GPU pour calculer la distance.
3. Nous démontrerons comment l'implémentation parallèle du K-means++ s'adapte aux grands ensembles de données et nous examinerons ses performances par rapport à celles du K-means++ séquentiel.

Le reste du chapitre est structuré en plusieurs sections. La section 2 donne un aperçu des travaux connexes sur l'accélération de l'algorithme du K-means++. La section 3 présente brièvement l'algorithme séquentiel du K-Means++. La section 4 présentera une description détaillée de notre implémentation parallèle originale de l'algorithme du K-means++ qui utilise les technologies OpenCL et SSE. La section 5 se concentrera sur la présentation des résultats de la synthèse et les discussions. Enfin, la section 6 conclura le chapitre.

4.2 **Traveaux connexes**

L'analyse des clusters est un sujet important pour les chercheurs du data mining depuis plusieurs années. Dans le passé, sous la condition d'une petite taille de données, les chercheurs se concentraient principalement sur le problème d'optimisation de l'algorithme du K-means lui-même.

Et puis plusieurs méthodes ont été inventées pour mettre en parallèle les algorithmes du clustering [20-22], y compris k-means [13-16, 23-25], afin d'améliorer leur efficacité encore plus.

Dans notre précédent article [15], nous avons comparé trois approches différentes traitant des implémentations parallèles du k-means sur CPU et GPU en utilisant les langages de programmations parallèles : OpenMP, Pthread, et OpenCL. Les résultats ont prouvé que les trois techniques parallèles ont montré une augmentation significative de la performance, où les meilleurs résultats sont obtenus par OpenMP pour les petits ensembles de données et par OpenCL pour les grands ensembles de données.

Cependant, peu de travaux ont été publiés sur la parallélisassions de l'algorithme du K-means++. Le premier travail est une thèse de Master réalisé par Karch [11], et le travail du Maliheh et al [26]. Dans [11], la thèse étudie la parallélisassions du k mean++ et K means sur

le GPU en utilisant CUDA. Ils ont seulement parallélisé le calcul des distances entre les points de données et différentes clusters et aucune des autres parties de l'implémentation du k-means++ n'a été parallélisée. Ils ont utilisé une carte GPU Nvidia GeForce 9600M GT pour obtenir une accélération maximale 5 fois plus rapide que celle de l'implémentation du CPU. Dans [26], les auteurs proposent de paralléliser les étapes les plus longues de l'algorithme du k-means++, l'étape d'initialisation est déchargée en parallèle vers le GPU en utilisant CUDA. Les centres initiaux sont choisis par une probabilité proportionnelle à la distance au centre le plus proche à l'aide de la carte GPU GeForce GTX 1070. Dans notre étude, nous avons parallélisé la phase d'affectation de données au niveau du GPU en utilisant OpenCL tandis que l'étape d'initialisation pour sélectionner les centroïdes initiaux a été exécutée en parallèle en utilisant les instructions SSE au niveau du CPU. La phase du recalculer les nouveaux K-centroïdes et la phase testsde la convergence ont été effectuées par le CPU car ces étapes ne consomment pas beaucoup de temps par rapport aux autres phases.

En résumé, nous avons adopté dans ce travail notre stratégie du parallélisassion K-means++ pour traiter de grands ensembles de données. Pour réaliser cette implémentation, nous avons utilisé la plate-forme OpenCL et la technologie SSE. Enfin, nous avons comparé cette implémentation parallèle du K-means++ avec l'implémentation séquentielle du K-means++.

4.3 **Méthode proposée**

4.3.1 **Implémentation séquentielle du Kmeans++**

L'algorithme du K-means [5] choisit les premiers centres au hasard. Puisqu'ils dépendent purement de la chance, ces centres peuvent être mal sélectionnés. L'algorithme K-means++ [12] tente de résoudre ce problème par une distribution uniforme des premiers centres. L'implémentation série du K-means++ est présentée dans le pseudo-code (Algorithme 1).

Algorithm 4.1: The Serial k-means++ Algorithm

Input: X; Set of data points (N), number_of_clusters(k), k>0

Output: A data points N partitioned into k clusters

1: Select centroids ***c1, c2, …, c:***

1-a. Choose one center ci uniformly at random from among the data points X.

1-b. Take a new center ci, For every data-point x, calculate **D(x),** the distance among x, and the nearest center that has just been picked. Choosing x € X with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$.

1-c. Repeat Steps 1-b until k centers have been chosen.

O(#N x D x K)

2: Iterate until a convergence condition issatisfied

3: For each data point ***xi,i=1..N:***

4: For each data cluster c*j, j=1..**K:***

5: Compute the distance to c*j, given all **D dimensions***

6: Assign the membership of data-point ***xi to nearest cluster j***

7: For each data cluster c*j, j=1..**K:***

8:Computethecentroid:***cj=themeanofalldata-points whose membership isJ***

O(N x D x K x #iterations)

We appeal the weighting utilized in step 1-b <D^2 weighting>.

L'algorithme 4.1 montre qu'après la sélection de chaque centre, la distance de chaque point par rapport au cluster le plus proche est mise à jour. Il est détaillé comme suit :

1-a Choisissez le premiercentre de façon uniforme et aléatoire parmi les points de données.

1-b Pour chaque point de données *x*, calculer D(*x*), la distance entre *x* et le centre le plus proche quivient d'être choisi .La distance euclidienne est couramment utilisée comme mesure pour la dispersion des clusters pour le clustering K-means++. Cette métrique est préférée car elle minimise la distance moyenne entre les points et les centroïdes [27].De plus, on choisit un nouveau point de données arbitraire comme nouveau centre en utilisant une distribution de probabilité pondérée où un point x est choisi avec une probabilité proportionnelle à$D(x)^2$voir formule (4.1).K-means++ utilise la méthode de pondération de la distance $D(x)^2$ pour sélectionner le prochain centre opportun. Cet algorithme réduit l'effet d'instabilité qui se produit dans K-means et fournit des résultats de regroupement plus stables [28].

LaProbabilité $p(x)=\frac{D(x)^2}{\sum_{x\in X} D(x)^2}$ (4.1)

1-c Répétezl'étape**1-b**jusqu'àceque *k*centresaientétéchoisis.

2- K-means++ est une amélioration de l'algorithme du K-means qui permet de surmonter la sélection arbitraire du centre initial des clusters. Nous pouvons continuer avec l'algorithme standard du K-Means après avoir initialisé les centroïdes. En utilisant l'algorithme du K-means++ pour initialiser les centroïdes, cela tend à améliorer la qualité des clusters. Bien qu'il soit coûteux en calcul par rapport à une initialisation aléatoire, K-Means convergent souvent plus rapidement. Après avoir choisi les premiers centres, il faut maintenant utiliser la méthode standard de clusteringdu K-means. Pour chaque itération, le clustering K-means se déroulera en deux phases ; tout d'abord, la phase d'attribution des données, qui associe chaque point de données au centroïde le plus proche en reposant sur une métrique de distance (ici la métrique de distance euclidienne est utilisée). Le résultat de la phase d'attribution des données est un vecteur d'appartenance indiquant le nouveau centre du cluster pour chaque point de données. Deuxièmement, après la phase d'attribution des données, k-means exécute une autre phase qui consiste à mettre à jour les centroïdes. Cette phase calcule les nouveaux centroïdes en utilisant la moyenne de tous les points de données dans chaque cluster. En supposant que le nombre d'objets (points) dans le cluster i est défini par s_i, la formule pour l'étape de mise à jour les clusters est indiqués ici :

$$c_i = \frac{1}{s_i}\sum_{j=1}^{s} X_j \quad (4.2)$$

Dans l'étape d'initialisation les K-centres initiaux, pour chaque centre de l'algorithme du K-means++, la complexité maximale du calcul est (N*K*D), où N est le nombre des points de données, D la dimensionnalité des données et K le nombre des clusters.L'étape de calcul de la distance est la partie la plus intensive de cette implémentation. Après l'étape d'initialisation, nous procédons àl'utilisation des k-means standard. Pour chaque itération de l'implémentation du k-means, la complexité maximale du calcul est (D*N*K +N*K + N*D). L'étape de calcul dela distance est la partie la plus complexe de cette implémentation.

La partie la plus gourmande en calculs de cette implémentation est l'étape du calcul de la distance qui nécessite une soustraction, une addition et une multiplication pour calculer la somme partielle de la distance pour chaque point de données. En général, le nombre d'opérations estapproximativement égal à *D*N*K*3 correspondant au nombre _d'itérations.

4.3.2 **ImplementationParallele du K-means++**

L'algorithme du K-Means++ est une méthode itérative. Ainsi, le calcul de la distance entre chaque point et les différentes clusters montre beaucoup de parallélisme dans chaque itération, puisque les boucles' For' des lignes de 3 à 6 (Algorithme 4.1) consomment beaucoup de temps, que ce soit pour les plus petits que pour les plus grands ensembles de données. Par conséquent, la partie de 3 à 6 des étapes du pseudocode de l'algorithme 1 est un excellent candidat pour explorer le parallélisme. Comme nous l'avons mentionné précédemment, nous cherchons à paralléliser la phase la plus chronophage de l'étape d'initialisation pour sélectionner les centres initiaux de l'algorithme K-means++, notamment la boucle 'For' de la ligne 1-b (Algorithme 1). La distance de tous les points au centre choisi doit être déterminée et ensuite, les clusters sont choisis sur la base d'une formule de probabilité. Ce sont les parties les plus éprouvantes pour la parallélisation de K-means++.

Une implémentation parallèle est présentée dans un pseudo-code< Algorithme 2 > en utilisant OpenCLetlatechnologie SSE. Les principales étapes sont décrites plus en détail ci- dessous. La figure 4.1 donne un aperçu de l'approcheproposée.

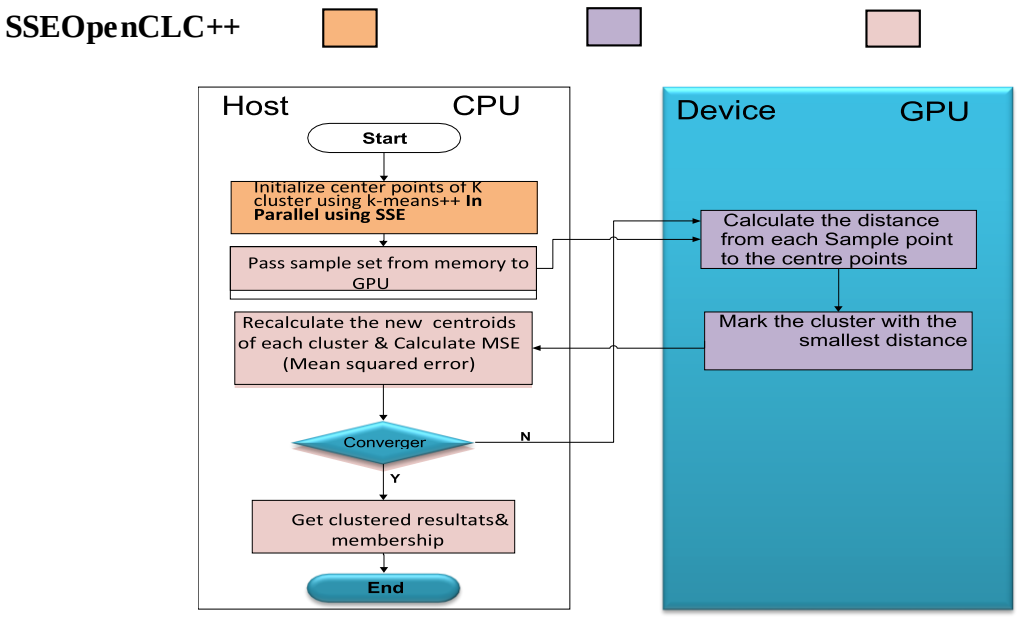


Figure 4-1: Étapes de l'implémentation parallèle du K-means++

La phase d'initialisation pour sélectionner les centroïdes initiaux est effectuée en parallèle à l'aide des instructions SSE (le code décrit en détail pour l'optimisation est présenté dans l'algorithme 4.2), et il convient ensuite de déterminer la distance de tous les points par rapport au centre choisi, puis les clusters sont sélectionnés sur la base d'une formule de probabilité (4.1).

Les instructions SSE nous permettent de calculer avec précision quatre nombres à virgule flottante [19, 29] ; nous avons déjà mentionné précédemment que la phase d'initialisation pour sélectionner les centroïdes initiaux doit être calculée avec précision, c'est donc un excellent choix pour effectuer des optimisations en utilisant la technologie SSE.

Ce travail étudiera la possibilité d'utiliser la technologie SSE pour accélérer l'étape d'initialisation de l'algorithme du K-means++ afin de sélectionner les centroïdes initiaux en exécutant plusieurs calculs avec une seule instruction. En 1999, avec le Pentium III, Intel a introduit la technologie SSE dans "l'architecture x86" [19]. Dans [19], les auteurs ont donné beaucoup de détails concernant les capacités et les motivations liées à la technologie SSE. Le SSE ajoute huit registres larges de 128 bits à l'architecture [19].

Les instructions SSE sont particulièrement utiles lorsqu'il faut exécuter des instructions sur différents éléments de données. Les vecteurs SSE sont des registres du 128 bits et nous permettent d'effectuer des calculs pour quatre types de données en virgule flottante simultanément. Nous avons accès à une grande quantité de puissance de calcul parallèle qui, autrement, irait inexploité. Parfois, l'utilisation de l'optimisation de SSE peut donner autant d'accélération en passant d'un processeur à un seul cœur à quadricoeur. Avec l'arrivée du Pentium 4, Intel a présenté la deuxième génération de SSE, généralement connue sous le nom de SSE2 [30].

Les instructions SSE, qui sont essentiellement des instructions à un seul flux et des données à multiples flux, réduisent donc la complexité du calcul de manière significative. Par exemple, dans notre implémentation dans la phase d'initialisation où nous calculerons les distances entre chaque paire de points de données et les clusters nécessitant "N" opérations en virgule flottante, dans la matrice de distance, un nombre similaire de distances peut être exécuté en N/4 cycles, au lieu de N cycles. En effet, quatre distances en virgule flottante doivent être effectuées en un seul cycle. Cependant, Il est nécessaire de réorganiser les données dans un format acceptable pour le calcul du SIMD. Cela consomme quelques cycles d'horloge.

Pour calculer la phase d'attribution de l'algorithme du K-means ++ en parallèle sur le GPU, la plateforme OpenCL [31] est utilisée comme environnement de programmation parallèle (voir Algorithme 4.2). OpenCL est un modèle de programmation parallèle qui tend vers un large éventail de frameworks : unités centrales de traitement CPU, unités de traitement graphique GPU, processeurs embarqués, processeurs de signaux numériques DSP, matériels reconfigurables FPGA, et combinaisons de ces systèmes. OpenCL est utilisé pour accélérer différentes applications dans divers domaines tels que la chimie, la bio-informatique et d'autres domaines de la recherche [31, 32].

Quand on utilise OpenCL, on est amené à penser à plusieurs méthodologies distinctes pour la parallélisation, parmi lesquelles :

- Mémoireglobale
- Mémoirelocale
- Mémoireprivée

Algorithm 4.2: A parallel K-Means++ Algorithm

```
Input: X; Set of Data points(N), number_of_Clusters(K), k > 0
Output: A data points N, partitioned into K clusters
// C, the list of k data points (vectors) that should be used as initial seeds for k-means++ clustering
 Function Parallel-K-means++_initialization_phase (X, k)
     for i from 0 to n − 1 do                          In Parallel using SSE
         D[i] := ∞
   —The initial seed is uniformly selected at random —
     C[0] := rand x ∈ X
   — The rest of the seeds are selected probabilistically by distance—
     for j from 1 to k − 1 do
         for i from 0 to n − 1 do                      In Parallel using SSE
             D[i] := min(‖X[i] − C[j − 1]‖, D[i])
           W[i] := D[i]
         i := WEIGHTED_RAND_ INDEX (W)
         C[j] := X[i]
  return C
// calculate the distance and membership in parallel using OpenCL
workgoup_size = NUM_CLUSTERS * NDIMS;
number_threads= NUM_POINTS ;
Function kmeans_kernel_assignment_phase (points, clusters, membership)
          g_id = get_global_id(0);
clusters_local[NUM_CLUSTERS*NDIMS] ;
          l_id=get_local_id(0) ;
clusters_local[l_id]  ⟶clusters[l_id] ;
barrier(CLK_LOCAL_MEM_FENCE)
    index=0
    for (int c=0 ;c<  NUM_CLUSTERS ; c++) do
             min_dist = ∞;
     for (int d = 0; d< NDIMS; d++) do
                 distance =0;
             ans=0 ;
                  ans = (points[d* NUM_POINTS + g_id] –  clusters_local [c * NDIMS + d]);
                  distance+ = (ans * ans) ;
              end
      if (distance <min_dist) then
                  min_dist = distance;
                  index = c;
       end
      end
             membership[g_id] = index;
  End
```

Les étapes de développement de l'algorithme parallèle K-means++ en utilisant OpenCL et SSE (voir figure 4.1) sont illustrées comme suit :

Étape 1 : Nous choisissons un cluster de manière uniforme et aléatoire parmi l'ensemble des données. Ensuite, nous utilisons les instructions SSE pour calculer la distance de tous les points au centre choisi qui doit être déterminé, les clusters sont choisis sur la base d'une probabilité. Ensuite, nous continuons et nous lisons les clusters et les données fournies par l'utilisateur, puis nous définissons le nombre de clusters. Le nombre prédéfini d'itérations est utilisé comme critère de convergence.

- Étape 2 : Nous copions l'ensembledes données et les K-clusters initiaux de la mémoire vers la mémoireglobale du GPU.
- Étape 3 : Dans cette étape, nous utilisons la plate-forme OpenCL pour calculer la distance euclidienne. Les N- Threads correspondent aux N-points des données, et la charge de travail de chaque Thread est exprimée par la formule (4.3). Ainsi, chaque cluster est chargé dans son groupe de travail une seule fois dans la mémoire locale, ensuite ilest partagéentre tous les threads du groupe de travail. Après cela, chaque thread permet de calculer la distance entre le point du l'ensemble de données et le cluster. Nous sauvegardons ensuite l'index du cluster correspondant auplus proche point de l'ensemble dedonnées.

$$\text{Thread} = \text{Ceil}\left(\frac{\text{Numberofobjects}}{\text{Threads}}\right) \quad (4.3)$$

- Étape 4 : Les données sont arrangées en fonction du centre le plus proche, puis les résultats partiels sont copiés dans la mémoire globale du GPU.
- Étape 5 : Dans le CPU, nous recalculons les nouveaux centroïdes et nous calculons l'erreur quadratique moyenne (MSE). Si le nombre d'itérations maximum est atteint, nous passons à l'étape suivante, sinon, nous retournons à l'étape précédente.

Étape 6 : À la fin de l'algorithme, nous obtenons les résultats du clusterings.

4.4 **Les résultats expérimentaux**

Dans cette section, nous expliquerons la partie évaluation et l'analyse des performances de l'algorithme du Kmeans++ en mode parallèle (GPU) et en mode série (CPU). Notre implémentation a été réalisée sous Windows7 et nous avons exploité notre environnement de calcul hétérogène, qui intégrait un Intel(R) Xeon(R) X5650 @2.67GHz, 12Go de RAM, ainsi qu'un GPU Nvidia GeForce GTX 1060 avec 6Go de mémoire, et la version Nvidia OpenCL 1.2 pour que le code GPU puisse exécuter K-means++.Nous avons utilisé des ensembles de

données générées de manière aléatoire pour réaliser nos expériences concernant les données en virgule flottante entre -1 et +1[33].

Nous avons étudié les performances des implémentations CPU et GPU respectives lors du traitement de données avec différentes tailles de clustersK, tailles d'objets N, tailles de dimensions D, et tailles d'itérations. La variable N représente le nombre d'objets qui varient de 2048 à 4194304 points de données qui sont calculés avec différentes dimensions. En outre, ces ensembles de données sont calculés pour différents clusters.

Dans ce travail, le nombre d'itérations défini précédemment est utilisé comme critère de convergence car de cette façon, il est beaucoup plus facile d'identifier le facteur d'accélération.

Pour une meilleure interprétation, les performances sont présentées à la fois par le débit enGFLOPS et le temps d'exécution enmillisecondes.

Nous avons examiné les performances de l'implémentation parallèle de l'algorithme K-means ++ à l'aide du GPU et celles de l'implémentation séquentielle du K-means ++ sur un CPU. Ensuite, pour les deux implémentations GPU et CPU, les mêmes ensembles de données ont été utilisés.

1. Nous avons comparé le temps de calcul uniquement dans la phase d'initialisation pour sélectionner les k-clusters initiaux en parallèle en utilisant les instructions SSE avec celui dans la phase d'initialisation pour sélectionner les K-clusters initiaux en série (sans utiliser SSE) de l'algorithme K-means ++, où le nombre d'itérations est le nombre de k-clusters.
2. Nous avons ensuite comparé le temps du clustering, qui est le temps global pour toutes les itérations, qui comprend le temps de calcul et de communication entre le CPU et le GPU et n'inclut pas le temps obtenu pour l'initialisation des K- clusters, où le nombre prédéfini d'itérations est utilisé comme critère de convergence. Lorsque le nombre d'itérations est fixe, il est beaucoup plus facile d'identifier le facteur d'accélération.

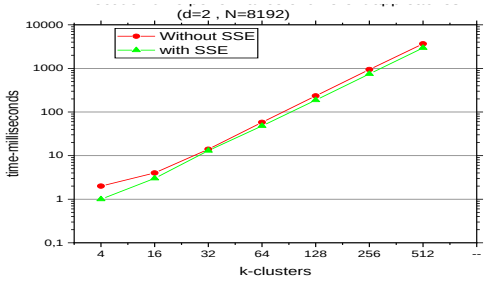


Figure 4-2:Temps d'exécution de l'étape d'initialisation avec et sans utilisation des instructions SSE du k-means++ avec différents tailles des clusters.

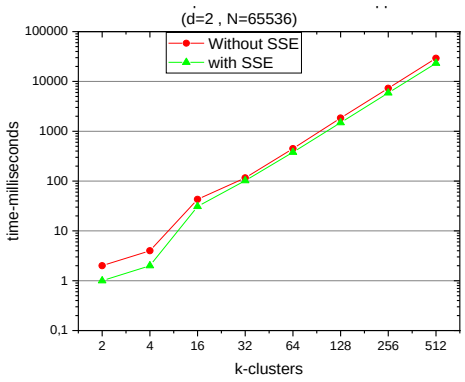


Figure 4-3: Temps d'exécution pour l'étape d'initialisation avec et sans utilisation des instructions SSE du k-means++ avec différents tailles des clusters

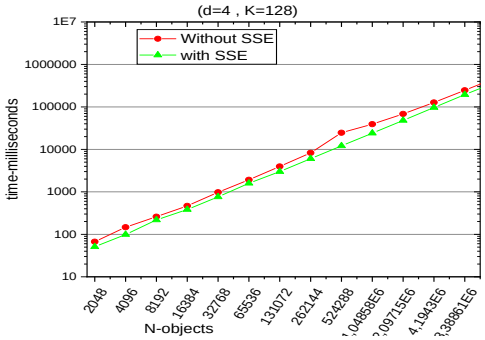


Figure 4-4: Temps d'exécution pour l'étape d'initialisation avec et sans utilisation des instructions SSE du k-means++ avec différents tailles d'objets et K-cluster =128

Nous avons appliqué les instructions SIMD SSE à l'étape d'initialisation (le code détaillé pour l'optimisation est indiqué dans l'algorithme 2) et obtenules résultats de l'optimisation qui sont présentés dans les figures4.2, 4.3 et4.4.

Les figures 4.2, 4.3 montrent le temps d'exécution de l'étape d'initialisationpour sélectionner les centroïdes initiaux en utilisant SSE et sans utiliser SSE de l'algorithme du k-means++ avec différentes tailles de clusterssur le CPUlorsque les points de données ont été fixés à 8192 et 65536, respectivement. On peut enconclure que, quel que soit lenombre de clusters, l'implémentation parallèle utilisant les instructions SSE surpasse l'implémentation séquentielle (sans utiliser leSSE).

La figure 4.4 montrent le temps d'exécution de l'étape d'initialisation pour la sélection des centroïdes initiaux lorsque nous avons utilisé l'SSE et sans utiliser l'SSE, de l'algorithme du K-means++ avec différentes tailles d'objets. On peut en conclure que, quel que soit le nombre d'objets, la version parallèle en utilisant l'SSE surpasse l'implémentation qui n'utilise pas l'SSE.

A partir de ces chiffres, nous constatons le gain de performance lors de l'utilisation de l'SSE. Dans la figure 4.4, nous avons pris un ensemble de données d'échelle de 524288 points avec 4 dimenssions pour calculer les 128 centres initiaux, le gain de performance atteint, grace à notre implémentation parallèle de l'étape d'initialisation en utilisant les instructions SSE estjusqu'à3X par rapport à l'absence d'optimisation (l'implémentation séquentielle de l'étape d'initialisation sans utiliser SSE pour l'algorithme du k-means ++).

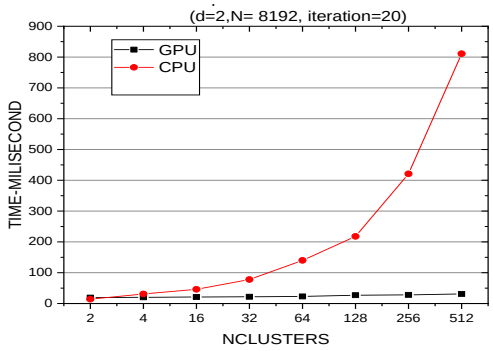


Figure 4-5:Temps d'exécution pour k-means++ avec différentes tailles des clusters sur le GPU et le CPU

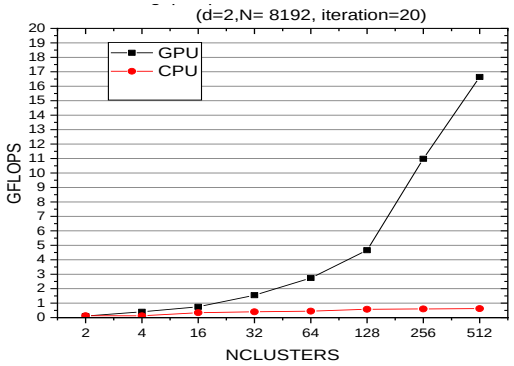


Figure 4-6: Débit pour k-means++ avec différentes tailles de clusters sur le GPU et le CPU

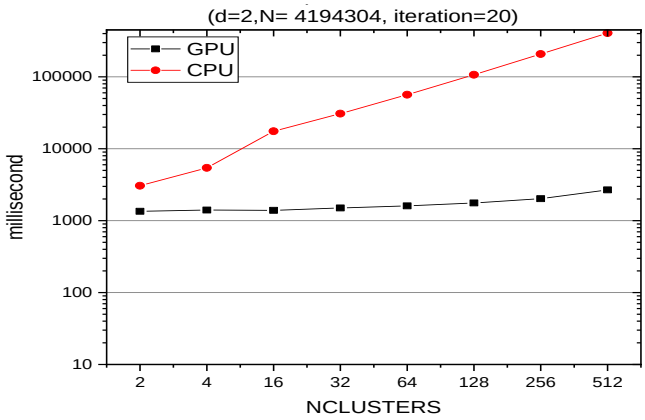


Figure 4-7: Temps d'exécution pour k-means++ avec différentes tailles du clusters sur le GPU et le CPU

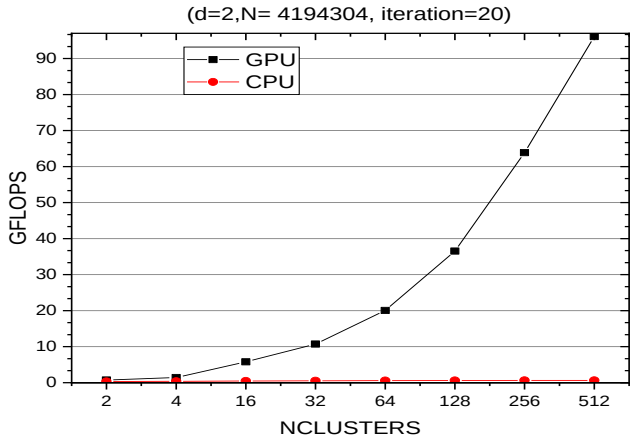


Figure 4-8: Débit pour k-means++ avec différentes tailles de clusters sur le GPU et le CPU

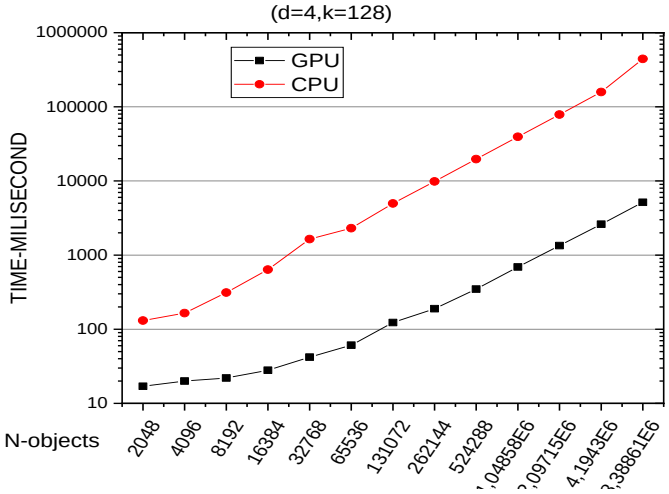


Figure 4-9: Temps d'exécution pour k-means++ avec différentes nombre d'objets sur le GPU et le CPU.

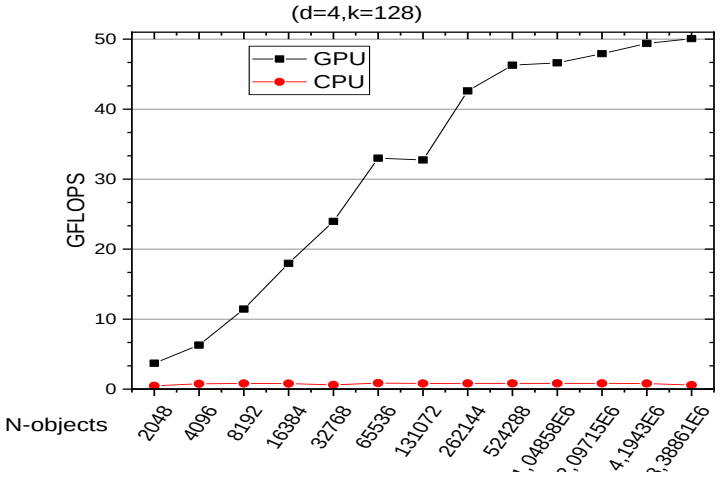


Figure 4-10: Débit pour k-means++ avec différentes nombre d'objets sur le GPU et le CPU.

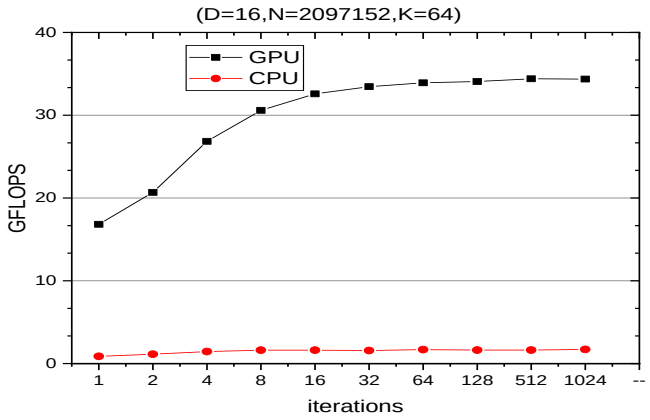


Figure 4-11: Débit pour k-means++ avec différentes nombre d'itérations sur le GPU et le CPU.

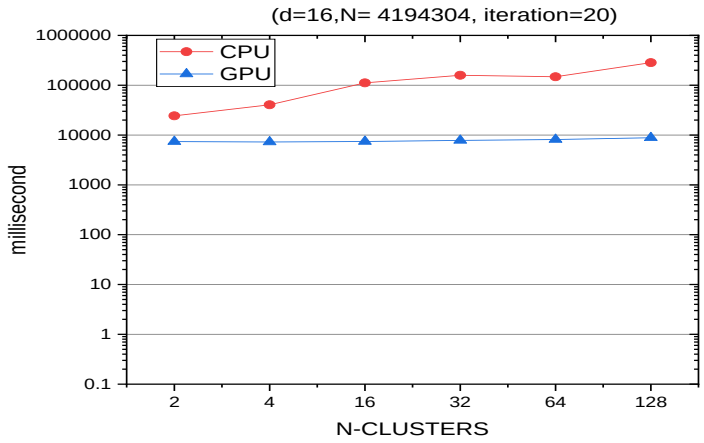


Figure 4-12: Temps d'exécution de k-means++ avec différentes nombre de clusters sur GPU et CPU pour l'ensemble de données de 4194304.

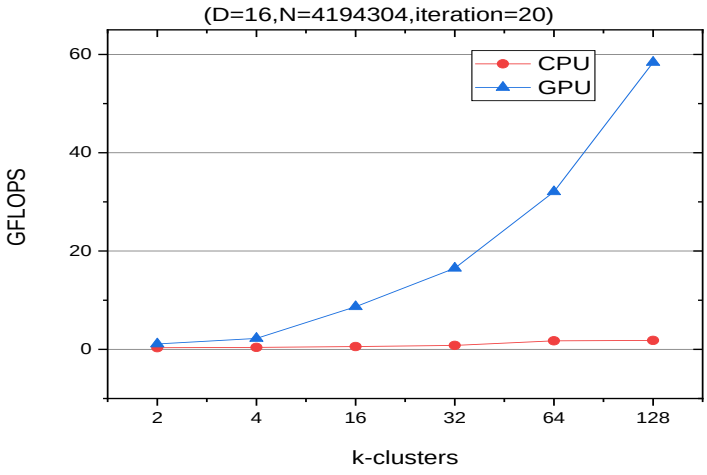


Figure 4-13: Débit pour k-means++ avec différentes nombre de clusters sur GPU et CPU avec dataset=4194304

Les débits d'exécution de différentes données dimensionnelles avec différents clusters, objets etd'itération sont présentés dans les figures 4.6,4.8, 4.10 ,4.11 et 4 . 13.

Les figures 4.5, 4.7, 4.9 et 4.12 montrent le temps d'exécution pour l'implémentation du K-means++ avec différentes nombres du cluster et avec des différents objets sur le GPU et le CPU. Plus précisément, le CPU surpasse l'implémentation GPU lorsque la taille des données est petite, le nombre de dimensionnalités est inférieur ou égal à 2 et le nombre de clusters est inférieur à 4 ($k<4$ et $d\leq 2$). De plus, l'implémentation GPU augmente la transmission des données car les ensembles de données sont copiés de la mémoire du CPU vers le GPU et ensuite les résultats partiels sont copiés du GPU vers la mémoire duCPU.

L'implémentation GPU du K-means++ permet d'obtenir de meilleurs résultats pour les grands nombres des clusters et les dimensions de taille moyenne, comme illustré dans les figures. Nous utiliseronslamémoire localeet la mémoire privée pour calculer la distance. Lorsque la taille du cluster est importante et du fait que les clusters sont enregistrés dans la mémoire locale, plus de données peuvent être réutilisées, ce qui améliore l'efficacité du noyau.

Certaines conclusions ont pu être tirées ; quelque soit le nombre d'objets, de clusters et d'itérations, le GPU surpasse le CPU de manière significative sauf lorsque si la taille des données est petite et que le nombre de dimensionnalités est inférieur ou égal à 2 et le nombre de clusters est inférieur à 4. Les performances du GPU sont cohérentes quelle que soit le nombre de clusters, d'objets ou d'itérations.

L'accélération de l'implémentation parallèle de l'algorithme du K-means ++ en utilisant le GPU par rapport à la version séquentielle du K-means ++ pourrait atteindre jusqu'à 152 fois.

Tableau 4-1: Implémentation du k-Means++ sur GPU et CPU : Résultats du débit maximal

Data size	Feature Dimension	K-clusters	Peak Throughput GPU (GFLOPS)	Peak Throughput CPU (GFLOPS)	Speed-Up
4194304	**8**	512	96,0843	0,635007	151,32
	16	256	85,928	0,934129	91,987
	32	64	26,8435	1,21758	22,047
	64	32	13,7424	1,36585	10,062
65536	**64**	16	7,80336	1,49241	5,2287
	128	4	4,88862	1,53684	3.181

Pour cette implémentation GPU, le débit maximal pour différents ensembles de données, dimensions et k-clusters est résumé dans le tableau 4.1. Le débit maximal de l'implémentation GPU est constamment élevé pour les dimensions 8 à 32, mais commence à

diminuer lorsque la dimension dépasse 32. Cela est dû au fait qu'il n'est plus possible d'utiliser complètement les ressources du GPU (mémoire locale) pour les grandes dimensions. Comme l'indique le tableau, l'accélération maximale est de 152 fois pour le traitement de données bidimensionnelles.

Comparaison avec le travail proposé et la littérature

Dans cette section, nous comparons notre meilleure performance obtenue avec d'autres meilleures performances de l'état de l'art, Comme le montre le tableau 4.2, nous obtenons une accélération de plus de X152 par rapport au Séquentiel k-means++. Nous avons testé une taille maximale de données de 16.777.216 tandis que le meilleur résultat publié à ce jour est obtenu par [26] qui utilise CUDA et obtient une accélération de X36 par rapport au Séquentiel k-means++. Ils ont testé une taille maximale de données de 10 millions.

Tableau 4-2: Comparaison entre le travail proposé et la littérature

Paper	Technology	Speed-Up
[11]	GPU Nvidia GeForce 9600M GT	5
[26]	GeForce GTX 1070	36
Ourwork	GPU NVIDIA GTX1060	152

4.5 **Conclusion**

Dans ce chapitre, nous avons présenté une étude complète sur le parallélisme de l'algorithme K-means++. Nous proposons une nouvelle implémentation parallèle de l'algorithme k-means ++ en utilisant l'unité de traitement graphique (GPU). La plateforme Open Computing Language (OpenCL) est utilisée comme environnement de programmation avec l'utilisation de la technologie Streaming SIMD Extension (SSE). Nous nous concentrons sur les optimisations directement ciblées cette architecture afin d'exploiter au mieux les capacités de calcul disponibles. Cet algorithme comprend trois étapes : l'initialisation, le traitement et la convergence. K means++ est une technique itérative à forte intensité de calcul ; chaque itération comprend deux phases : l'attribution des données et la mise à jour des k centroïdes qui permet de recalculer les nouveaux centroïdes. Pour accélérer les parties intensives en calcul de k-means++, l'étape d'initialisation est calculée en parallèle en utilisant la technologie SSE et l'étape de l'attribution des données est chargée sur le GPU. Seule l'étape de mise à jour des K centroïdes et l'étape de test de convergence sont effectuées par le CPU. Nos résultats montrent que l'implémentation ciblée d'architecturesparallèles hybrides (CPU & GPU) est la plus appropriée pour les grandes tailles de données. Nous avons atteint un débit 152 fois supérieur à celui de la version séquentielle de K-means ++.

Dans le chapitre 5, nous nous concentrons sur le développement d'une méthode de détection d'objets abandonnés dans la surveillance visuelle. En raison de l'aspect critique d'un tel système, la robustesse et le coût de calcul sont nos principales préoccupations. Le système proposé est capable de détecter les objets abandonnés, tel que les bagages dans les zones de transit et les objets immobiles de toutes formes dans les environnements intérieurs et extérieurs, avec des changements des lumières brusques et aléatoires, et dans les zones encombrées. La nouveauté de notre méthode est l'utilisation de l'algorithme du clustering k-means pour obtenir le background initial ou chaque pixel est représenté par un cluster et les contours au lieu des pixels dans la détection de régions statiques. le temps d'exécution du K-mean traditionnel a été amélioré en proposons un algorithme parallèle kmeans en utilisant la plateforme OpenCL. Deux scores robustes sont également utilisés pour la classification des objets abandonnés.

4.6 **Réferences**

[1] Han, J., Kamber, M. (2006). Data Mining: Concepts and Techniques. Second Edition, Elsevier Inc., Rajkamal ElectricPress.doi: 10.1016/C2009-0-61819-5.

[2] Estivill, C.V. (2002). Whysomanyclusteringalgorithms-a position paper. In: Newsletter ACM SIGKDD Explorations Newsletter Homepage Archive, 4(1): 65-75.https://doi.org/10.1145/568574.568575.

[3] Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. Proc NatlAcadSci., 95(25):14863- 14868. doi: 10.1073/pnas.96.19.10943-c.

[4] Cuomo, S., Michele, P., Pragliola, M. (2017). A computationalscheme to predictdynamics in IoTsystemsbyusingparticlefilter.ConcurrComput,29(11): 4101.https://doi.org/10.1002/cpe.4101

[5] MacQueen,J.B.(1967).SomeMethodsforclassification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on MathematicalStatistics andProbability.Berkeley,UniversityofCaliforniaPress, 1:281-297.https://www.bibsonomy.org/bibtex/25dcdb8cd9fba78e0e791af619d61d66d/enitsirhc.

[6] Selim, S.Z., Ismail, M.A. (1984). K-means type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(1): 81-

87.https://doi.org/10.1109/TPAMI.1984.4767478

[7] Arthur, D., Sergei, V. (2007). K-means++: The advantages of carefulseeding. Proceedings of the EighteenthAnnual ACM-SIAM Symposium onDiscreteAlgorithms,SODA2007,NewOrleans,Louisiana,USA, 1027-1035.https://doi.org/10.1145/1283383.1283494

[8] Lee, S.S., Won, D., McLeod, D. (2008). Tag-geotagcorrelation in social networks. In Proceedings of the 2008ACMWorkshoponSearchinSocialMedia,59-66. https://doi.org/10.1145/1458583.1458595

[9] Inouye, D. (2010). Multiple post microblogsummarization. REU Research Final Report, 1:34-40.

[10] Velardi, P., Navigli, R., Cucchiarelli, A., D'Antonio, F. (2008). A new content-based model for social network analysis. In 2008 IEEE International Conference on Semantic Computing, pp. 18-25.
https://doi.org/10.1109/ICSC.2008.30

[11] Karch, G. (2010). GPU basedacceleration of selectedclustering techniques. Department of Electrical and Computer Engineering and Computer Sciences, SilesianUniversity of Technology in Gliwice, Silesia,Poland.

[12] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S. (2012). Scalable k-means++. Proceedings of the VLDB Endowment, 5(7):622-633.doi: 10.14778/2180912.2180915.

[13] Zhang, J., Wu, G., Hu, X., Li, S., Hao, S. (2011). December. A parallel k-means clustering algorithm with mpi. In Fourth International Symposium on Parallel Architectures, Algorithms and Programming, 60-64.doi: 10.1109/PAAP.2011.17.

[14] Soua, M., Kachouri, R., Akil, M. (2018). GPU parallelimplementation of the new hybridbinarizationbased on Kmeansmethod (HBK). Journal of Real-Time Image Processing, 14(2): 363-377.
https://doi.org/10.1007/s11554-014-0458-2

[15] Daoudi, S., Zouaoui, C.M.A., El-Mezouar, M.C.,Taleb,N. (2019). A comparative study of parallel CPU/GPU implementations of the K-MeansAlgorithm. In 2019 International Conference on Advanced Electrical Engineering (ICAEE), pp. 1-5.https://doi.org/10.1109/ICAEE47123.2019.9014783

[16] Clemens, L., Sebastian, B., Steffen, Z., Volker, M., Tilmann, R. (2018). Efficient k-

Means on GPUs. publication rightslicensed to the Association for ComputingMachinery. ACM.doi: 10.1145/3211922.3211925.

[17] Khronos, G. (2017). OpenCL - The open standard for parallelprogramming of heterogeneoussystems. https://www.khronos.org/opencl/.

[18] Stone, J.E., Gohara, D., Shi, G. (2010). OpenCL: A parallelprogramming standard for heterogeneouscomputingsystems. Computing in Science & Engineering, 12(3): 66.https://doi.org/10.1109/MCSE.2010.69

[19] Thakkur, S., Huff, T. (1999). Internet streaming SIMD extensions. Computer, 32(12): 26-34.
https://doi.org/10.1109/2.809248

[20] Raymound, N.T., Han, J. (1994). Efficient and Effective clusteringmethods for spatial data mining. In Proceedings of VLDB:144-155.https://dl.acm.org/doi/10.5555/645920.672827.

[21] Zhang,T.,Ramakrishnan,R.,Livny,M.(1996).BIRCH: an efficient data clusteringmethod for very large databases. ACM Sigmod Record, 25(2): 103-114. https://doi.org/10.1145/235968.233324

[22] Wang,M.,Zhang,W.,Ding,W.,Dai,D.,Zhang,H.,Xie, H.,Xie,J.(2014).Parallelclusteringalgorithmforlarge- scalebiological data sets. PloS One, 9(4): e91315. https://doi.org/10.1371/journal.pone.0091315

[23] Tang, Q.Y., Khalid, M.A. (2016). Acceleration of K- meansalgorithmusingAltera SDK for OpenCL. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 10(1): 1-19.https://doi.org/10.1145/2964910

[24] Cuomo, S., De Angelis, V., Farina, G., Marcellino, L., Toraldo,G.(2019).AGPU-acceleratedparallelK-meansalgorithm.Computers&ElectricalEngineering,75:262-274.https://doi.org/10.1016/j.compeleceng.2017.12.002

[25] Li, Y., Zhao, K., Chu, X., Liu, J. (2013). Speeding up k- meansalgorithm by gpus. Journal of Computer and System Sciences, 79(2): 216-229.https://doi.org/10.1016/j.jcss.2012.05.004

[26] Maliheh, H.S., Reza, T. (2019). Parallelization of Kmeans++ using CUDA. (inpress).

[27] Singh, A., Yadav, A., Rana, A. (2013). K-meanswithThreedifferentDistanceMetrics.InternationalJournalof Computer Applications, 67(10).doi:10.5120/11430-6785.

[28] Zhang, M., Duan, K.F. (2015). Improvedresearch to k- means initial cluster centers.

In 2015 Ninth International Conference on Frontier of Computer Science and Technology, pp. 349-353.https://doi.org/10.1109/FCST.2015.61

[29] Diefendorff, K. (1999). Pentium iii= pentium ii+ sse. Microprocessor Report, 13(3):1-6.

[30] Sager, D. (2001). Desktop Platforms Group, and Intel Corp. The microarchitecture of the Pentium 4 processor. Intel TechnologyJournal.

[31] Kussmann, J., Ochsenfeld, C. (2017). Employingopencl toaccelerateabinitiocalculationsongraphicsprocessingunits. Journal of Chemical Theory and Computation, 13(6): 2712-2716.https://doi.org/10.1021/acs.jctc.7b00515

[32] Cadenelli, N., Jaksić, Z., Polo, J., Carrera, D. (2019). Considerations in usingOpenCL on GPUs and FPGAs for throughput-orientedgenomicsworkloads. Future Generation Computer Systems, 94:148-159.https://doi.org/10.1016/j.future.2018.11.028

[33]http://www.cs.virginia.edu/~kw5na/lava/Rodinia/Packages/Current/rodinia_3.0/data/km

CHAPITRE 5 :
technique robuste basée sur parallèle clustering K-means pour la détection AO.

Sommaire

5.1 Introduction

Au cours des dernières années, le développement des systèmes de surveillance visuelle a connu une croissance exponentielle. Les parcs nationaux, les aéroports, les stations de métro, les rues, les campus et les lieux publics sont tous surveillés. Ces systèmes sont utilisés pour surveiller les lieux publics, les résidences privées, les zones industrielles, etc. voir figure 5.1.

L'alimentation des caméras est généralement surveillée par un opérateur humain, qui effectue une vérification des lieux et cherche à détecter les événements anormaux. L'opérateur humain peut échouer dans sa tâche, en raison de ses capacités limitées à maintenir une surveillance continue. De plus, les grands réseaux de caméras CCTV ont donné lieu à des flux énormes et ont augmenté le nombre d'opérateurs humains pour effectuer la surveillance. Tous ces facteurs ont conduit au développement des systèmes de vidéosurveillance automatisés, capables de traiter plusieurs flux en même temps et de permettre la détection d'événements en temps réel. La communauté scientifique dans le domaine de la vision par ordinateur cherche à développer des solutions intelligentes et automatisées pour la détection d'événements visuels.

Dans ce but, de nombreuses techniques ont été utilisées pour l'analyse de la vidéo, telles que la détection d'objets en mouvement, le suivi d'objets et l'identification de personnes.

Dans ce chapitre, nous nous concentrons sur l'inspection d'objets physiques, plus précisément, l'objectif de ce travail est la détection de bagages abandonnés dans des lieux publics. Les attaques terroristes ont été exécutées en utilisant des objets explosifs camouflés dans des bagages, et abandonnés dans des lieux publics (zones de transit). La figure 5.2 montre un exemple de cas d'objet abandonné. Nous concentrons sur le développement d'une méthode de détection d'objets abandonnés dans la surveillance visuelle. En raison de l'aspect critique d'un tel système, la robustesse et le coût de calcul sont nos principales préoccupations. Le système proposé détecte les objets abandonnés, comme les bagages dans les zones de transit et les objets immobiles de toutes formes dans les environnements intérieurs et extérieurs, avec des changements des lumières brusques et aléatoires, et dans les zones encombrées.Ce chapitre vise pour son essentiel la conception et la mise en œuvre d'algorithme parallèle temps réels de clustering réputés d'être très chronophage et énergivore dans la détection des objets abandonnés ciblant les dispositifs matériels les plus communs dans les développements d'applications de vidéosurveillance sur des systèmes embarquées hétérogènes.Plus précisément, l'objectif du système étant de délimiter avec précision les frontières des objets pour les classifiers comme des objets abandonnés, K-means jouent un rôle crucial pour leur capacité à séparer clairement les différentes zones. La nouveauté de notre méthode est l'utilisation de l'algorithme parallèle du clustering K-means avant la soustraction du background ou chaque pixel est représenté par un cluster ensuite on utilise les contours au lieu des pixels dans la détection de régions statiques.Deux scores robustes sont également utilisés pour la classification des objets abandonnés [1]. Le principal inconvénient est la complexité de calcul de l'algorithme du k-means. Ce chapitre décrit le développement de l'algorithme de clustering K-means parallèle, afin de fournir un traitement en temps réel pendant les procédures de détection. Le temps d'exécution du K-means traditionnel a été amélioré en proposons un algorithme parallèle K-means en utilisant la plateforme OpenCL. Le traitement doit être effectué par le GPU et le CPU. A la fin, l'objet détecté doit être affiché et une comparaison de temps doit être faite entre les sorties du CPU et du GPU. Le graphique de comparaison de temps doit être tracé.

Figure 5-1: : Exemples du déploiement de caméras CCTV dans les lieux publics, aéroports, parcs, rues

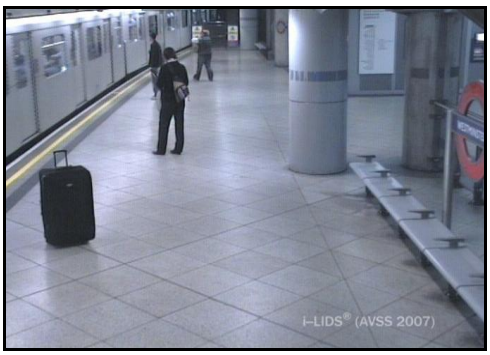


Figure 5-2: Exemple d'une situation de bagage abandonné

5.2 Travaux connexes

La détection automatique des objets abandonnés est un problème important pour la vidéosurveillance intelligente, compte tenu de son impact considérable sur la sécurité dans les lieux publics. Un tel système doit détecter les objets déposés par des personnes dans la zone

surveillée. De nombreux travaux ont été consacrés à la détection d'objets abandonnés au cours des dernières décennies.

La plupart des travaux de détection d'objets abandonnés utilisent la soustraction de l'arrière-plan comme étape préliminaire de bas niveau[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] pour détecter les régions ou les objets du premier plan, bien que Smith et al. [16] commencent directement par le suivi d'objets multiples dans la scène en utilisant la méthode du Markov Chain Monte Carlo (MCMC). Après la soustraction de l'arrière-plan, certains travaux visent à réduire les faux positifs en utilisant les méthodes de suivi et de classification des objets [2], [3], [8], [9], [10], [11], [12], tandis que d'autres approches utilisent la détection des contours [1], [17] ou des modèles génératifs [7]. Certains modèles traitent le problème de la détection des objets abandonnés en utilisant des automates à états finis [10], [13], [14], tandis que d'autres utilisent l'inférence basée sur la logique temporelle comme solution alternative [2], [6], [18].Contrairement à tous les autres travaux, Kong et al [19] essayent de détecter les objets abandonnés sur la route en utilisant une caméra mobile. Ferryman et al. [18] présentent un algorithme qui combine le concept de propriété avec la reconnaissance automatique des relations sociales afin de détecter l'abandon d'objets. Pham et al. [15] proposent une méthode en deux étapes pour la détection des objets abandonnés. La première étape tente de détecter tous les objets abandonnés possibles, en évitant les faux négatifs. Dans la deuxième étape, leur méthode réduit les fausses alarmes en utilisant la correspondance de similarité entre les candidats de premièr plan et le modèle d'arrière-plan.

L'une des méthodes les plus utilisées pour la soustraction de l'arrière-plan est le clustering, les structures de cluster considèrent que chacun des pixels de l'image peut être représenté temporellement par des clusters [20].

En 2005, Butler et al. [21] ont proposé une approche qui représente chacun des pixels de l'image par un ensemble de clusters. Les clusters sont arrangés en fonction de la possibilité qu'ils construisent l'arrière-plan et sont entraînés à traiter les changements d'arrière-plan et d'illumination. Les nouveaux pixels sont associés à l'ensemble des clusters et sont classés selon l'appartenance du cluster correspondant à l'arrière-plan. Pour améliorer la robustesse, Duan et al [22] ont utilisé en 2011 une approche génétique K-means. Le concept permet d'atténuer les inconvénients du K-means conventionnel qui présente l'aspect randomisé, ce qui entraîne des lacunes dans l'optimisation globale.

En 2014, Sinha et al. [23], une nouvelle segmentation vidéo effectue une décomposition des images entre arrière-plan et premier plan. Dans cette étude, un mélange avec un filtre mean-

shift simplifié et le clustering K-Means sont utilisés pour modéliser l'arrière-plan. Le principal innovaient du k-means c'est la compelexitè de temps de calcul.

En 2017, les auteurs [24], ont utilisé le calcul parallèle de l'algorithme du k-means pour l'extraction d'un modèle d'arrière-plan. La plateforme utilisée c'est Pthread .Ils ont amélioré le temps d'exécution du K-meanssséquentiel pour obtenir un arrière-plan initial, le temps a été minimisé à 50% par rapport au conventionnel K-means. Dans la littérature, il existe différents travaux concernant le parallélisme du l'algorithme K-means.

La parallélisation de l'algorithme K-means est le sujet de divers travaux et articles. Indépendamment de la technique utilisée, que ce soit sur des GPU [25], [26], [27], sur l'architecture Intel MIC (Many Integrated Core) [28], sur des clusters [29], sur des FPGA [30] ou autres.

Les travaux de Baramkar et al. [31] ont étudié les différentes implémentations du K-means parallèles basés sur les GPU, mais ils sont concentrés uniquement sur la classification générale, sans prendre en compte la haute dimensionnalité des données, les datasetsutilisés dans ce travail sont différents datasetsutilisés dans notre travail. Donc il est difficile d'effectuer des comparaisons directes avec ces travaux.

Zechner et al. [32] ont proposé une implémentation parallèle de cet algorithme en utilisant à la fois le CPU et le GPU. En particulier, le GPU a été utilisé uniquement pour le calcul de la distance, tandis que la mise à jour des centroïdes a été laissée au CPU, dans notre travail toutes les étapes du clustering sont effectuées par le GPU. Ils ont classé un ensemble de données artificielles avec des éléments bidimensionnels allant de 500 à 500 000.

Une approche similaire est présentée dans [33],[34], la différence étant que la mise à jour des clusters a également été effectuée sur le GPU. Cependant, entre le calcul de la distance et la mise à jour des centroïdes, ils ont effectué un traitement sur l'hôte pour la mise à jour de chaque étiquette de pixel. Ce choix conduit à une perte de temps, nous avons déplacé tous les calculs du côté du GPU.

Les auteurs Baydoun et al. [35] ont développé un K-means parallèle pour la classification des images RVB, comme Ils ont adopté comme métrique une simple distance cartésienne et n'ont parallélisé que ce calcul.

Les auteurs de Lutz et al. [36] ont proposé une implémentation parallèle du K-means en utilisant un GPU NVIDIA GTX 1080. Ils n'ont réalisé que des expériences produisant quatre groupes, mais aucun détail n'est fourni dans l'article concernant les datasetsutilisées.

Dans notre précédent article [26], nous avons comparé trois approches différentes traitant des implémentations parallèles du k-means sur le CPU et le GPU en utilisant les langages de programmations parallèles : OpenMP, Pthread, et OpenCL. Les résultats ont prouvé que les trois techniques parallèles ont montré une augmentation significative de performance, où les meilleurs résultats sont obtenus par OpenMP pour les petits ensembles de données et par OpenCL pour les grands ensembles de données.

5.3 **Méthode Proposée**

Nous utilisons un modèle simple mais efficace pour décrire la détection d'un objet abandonné. La nouveauté de notre méthode est l'utilisation de l'algorithme du clustering k-means parallèle avant la soustraction du background ou chaque pixel est représenté par un cluster, k-means doit être effectué par le GPU a l'aide de la frameworkOpenCL et par le CPU en mode serial. La méthode utilise les contours pour détecter et classer les objets abandonnés AO [1]. Nous nous concentrons sur la réduction des fausses alarmes. Nous détectons les contours stables en appliquant une accumulation temporelle sur la sortie d'une méthode de détection des contours de premier plan. Ensuite, afin de classer les candidats objets abandonnés (AO), nous utilisons un score [1] robuste basé sur la configuration et l'orientation des contours des candidats AO pour vérifier si la boîte englobante entoure réellement un objet ou non. Un autre score [1] basé sur la consistance des contours statiques est utilisé pour vérifier la stabilité de l'objet dans le temps, et pour rejeter les candidats présentant des mouvements faibles et internes, comme les personnes immobiles. Les deux figures 5.3 et 5.4 illustre les principales étapes du système proposé (version séquentielle et version parallèle OpenCL). A la fin, l'objet détecté doit être affiché et une comparaison de temps doit être faite entre les sorties du CPU et du GPU, et le graphique de comparaison de temps doit être tracé.

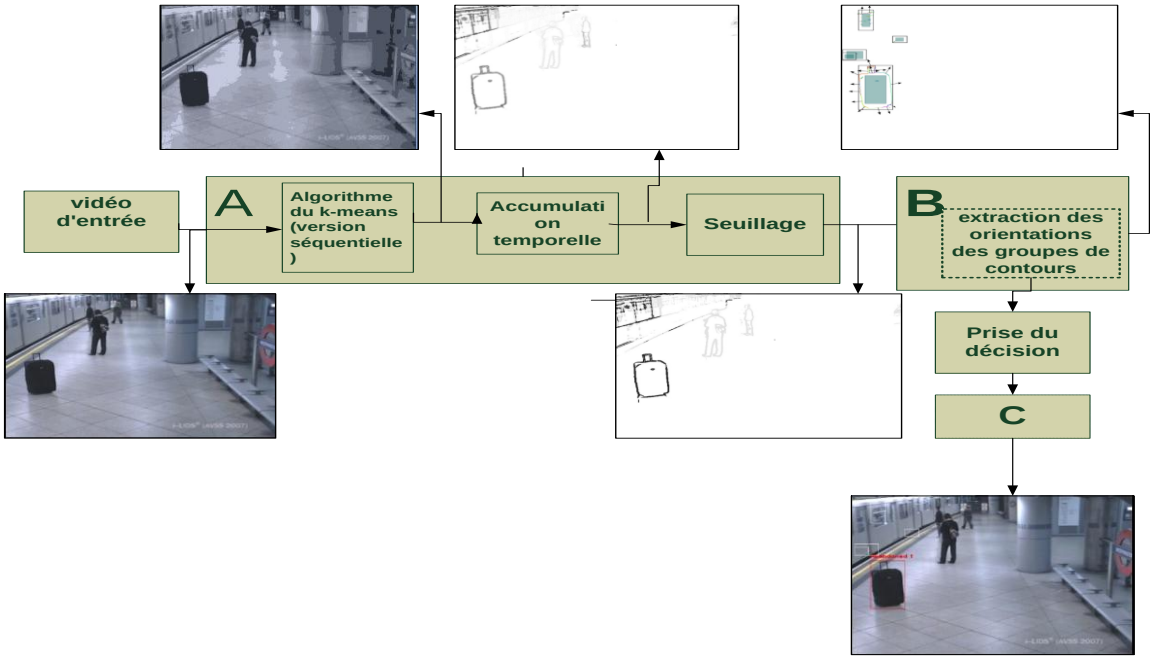


Figure 5-3: Diagramme de la méthode proposée. A : détection des contours stables en appliquant K-means (version séquentielle). B : extraction des orientations. C : Classification des candidats AO.

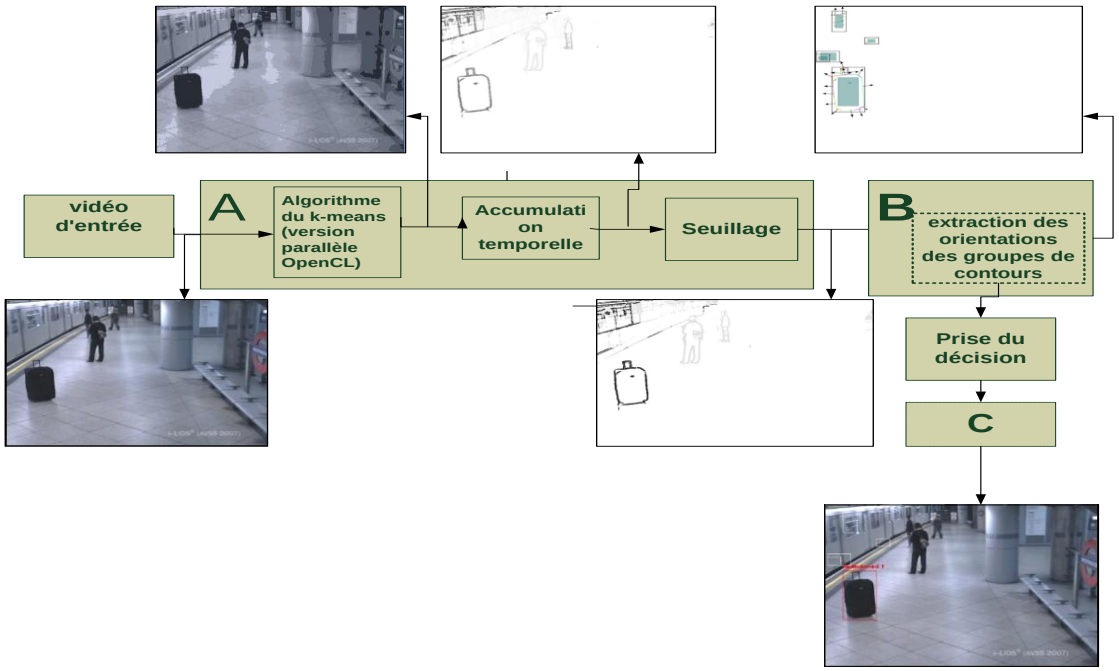


Figure 5-4: Diagramme de la méthode parallèle proposée. A : détection des contours stables en appliquant K-means (version parallèle OpenCL). B : extraction des orientations. C : Classification des candidats AO.

5.3.1 **La détection des objets en mouvement**

5.3.1.1 **Algorithme K-means version séquentielle :**

K-means (Mac Queen, 1967) [37]est l'un des algorithmes d'apprentissage non supervisé les plus simples qui répond à la technique bien connue du clustering couramment utilisée, qui est une méthode simple et rapide. Les étapes de la méthode sont les suivantes [38] :

Algorithm 5.1 : Algorithme séquentiel du K-means
Entrée : donné dataset (Tableau_Pixels), définir le nombre K souhaitable de clusters.
Sortie : Affectation de chaque donnée aux clusters.
Début : Étape 1. choisir aléatoirement les K cluters parmi tous les pixels du dataset Etape 2. Affecter pour chaque pixel i au cluster le plus proche Étape 3. Recalculer le centre du cluster en faisant la moyenne de tous les pixels. Etape 4. Refaire les étapes 2 et 3 jusqu'à ce qu'une convergence soit atteinte (par exemple, le centre de la grappe reste inchangé ou le nombre d'itération est achevé). **Fin**

5.3.1.2 **L'algorithme du k-means version Parallèle en utilisant OpenCL :**

L'algorithme parallèle est présenté dans le pseudocode Algorithme 5.2. Cette implémentation est plutôt simple, la phase d'affectation des données aux clusters le plus proche et la phase de recalculer les nouveaux K-centroïdes sont déchargées vers le périphérique OpenCL (effectuer le traitement en parallèle par le GPU), seulement la phase d'initialisation des k-clusters et la phase de tests la convergence sont effectuées en mode séquentiel par le CPU hôte (voir la Figure 5.5).

Le workflow de l'algorithme parallèle utilise deux noyaux :

Le premier noyau permet de calculer la phase d'affectation des données en parallèle en utilisant la mémoire locale ; mettre les données dans le registre et les clusters dans la mémoire locale.Le deuxième noyau permet de calculer en parallèle de la phase de recalculer les nouveaux K-centroïdes ; elle est composée de deux sous-phases illustrées en pseudo-code (Algorithme5.2) :Premièrement, le calcul de la somme temporaire des caractéristiques et deuxièmement la somme temporaire de masse de chaque cluster. Les centroïdes finaux sont obtenus en divisant les vecteurs de somme de caractéristiques finales par le vecteur de somme

de la masse finale, cette division est exécutée par le CPU. Tout d'abord, en divisant en groupes les N données (N/taille), le conflit d'écriture diminue, puisque chaque groupe écrit son propre temporaire (le FeatureSum et le Mass Sum) et n'a aucune influence sur les autres groupes. Enfin, nous calculons le FeatureSum et le Mass Sum final sur le CPU et les clusters finaux sont obtenus en divisant les vecteurs du FeatureSum final par le vecteur du Mass Sum final. Il est important de souligner que la « **taille »** dans l'algorithme 5.2, qui devrait être un multiple du nombre d'unités de calcul (CU) de manière à garantir une efficacité de programmation importante sur le GPU.

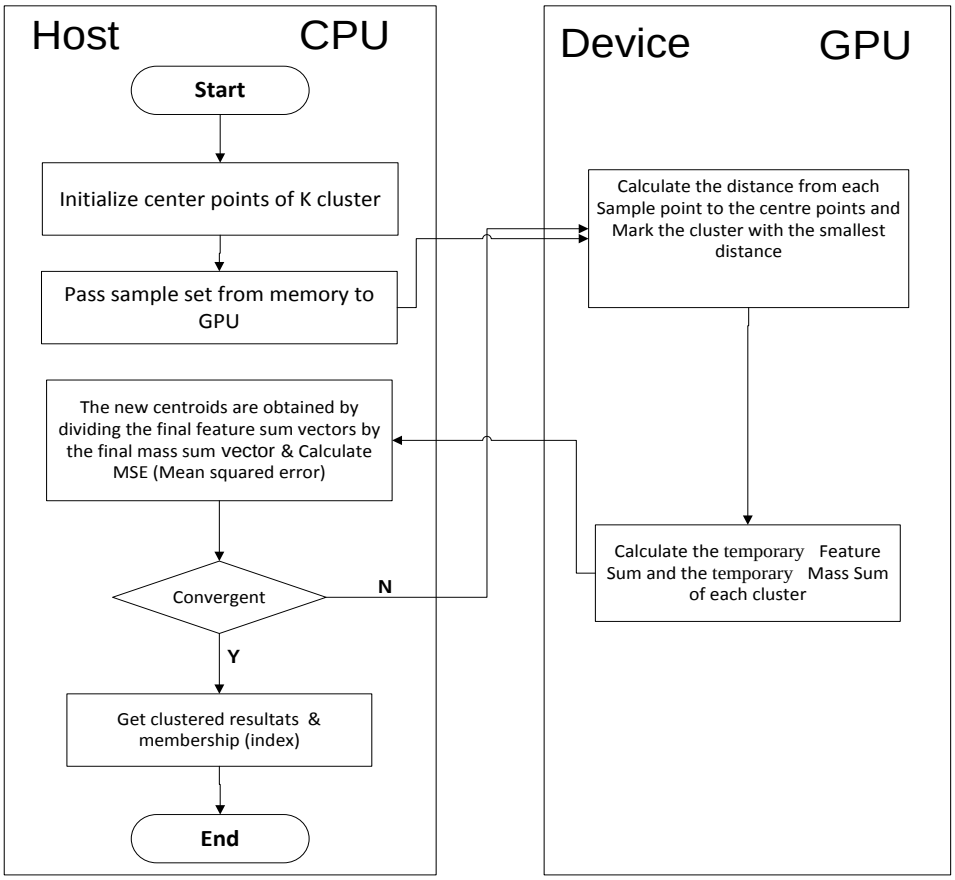


Figure 5-5: Étapes de l'algorithme K-means version parallèle en utilisant OpenCL.

Algorithm 5.2 : l'algorithme K-means version parallèle.

```
define NUM_POINTS ;
define NUM_CLUSTERS ;
define NUM_FEATURES ;

kernelcompute_distances (feature, clusters, membership)
        g_id = get_global_id(0);
        index=0
    if (g_id< NUM_POINTS ) then
        min_dist = ∞;
        for (int k = 0; k<NUM_CLUSTERS; k++) do
                distance=0 ;
                        ans=0 ;

for(int i=0 ; i< NUM_FEATURES ;i++) do
                ans = feature[i* NUM_POINTS + g_id] – clusters [k* NUM_FEATURES + i];
                distance+ = (ans * ans) ;
end
if (distance <min_dist) then
        min_dist = distance;
                index = k;
        end
      end
membership[g_id] = index;
End kernel


kernel   compute_new_clusters  (points , Feature_Sum_temp,  Mass_Sum_temp, membership, size)
size: is is the multiple of the number of CU;
set cluster_groupe [ NUM_CLUSTERS * NUM_FEATURES]=0.0f;
set cluster _size [NUM_CLUSTERS]=0;
 p_id = id*size ;
 id = get_global_id(0);
 t_fm = points + (p_id* NUM_FEATURES);
        for(int i=0;i<size ;i++) do
                if (p_id+i < NUM_POINTS ) then
                        cluster_id = membership[p_id+i];
                        for(int k=0;k< NUM_FEATURES ;k++) do
                 cluster_groupe [cluster_id * NUM_FEATURES + k] += t_fm[i* NUM_FEATURES +k];
                        end
                          cluster_size [cluster_id] += 1;
                end
        end
        write   cluster_groupe  to  Feature_Sum_tempglobal memory ;
            write   cluster_sizeto Mass_Sum_temp global memory ;
End kernel
```

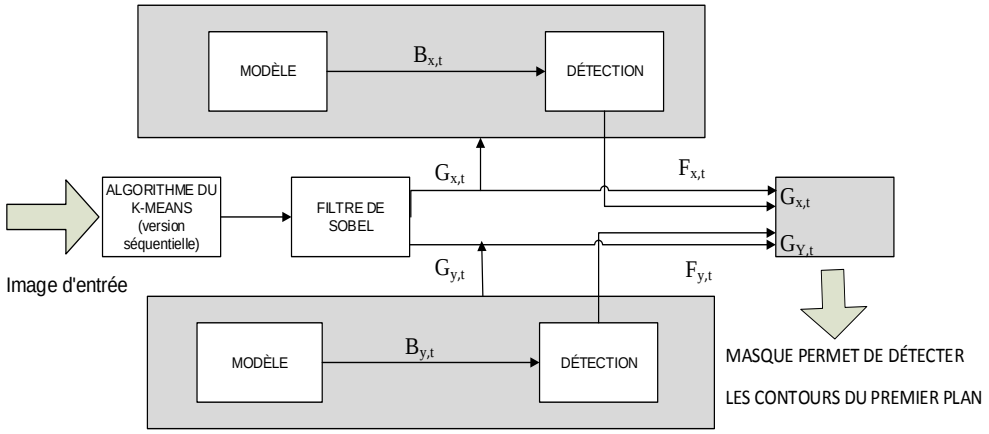


Figure 5-6: Diagramme de blocs de la méthode de soustraction de l'arrière-plan basée sur les contours (version séquentielle).

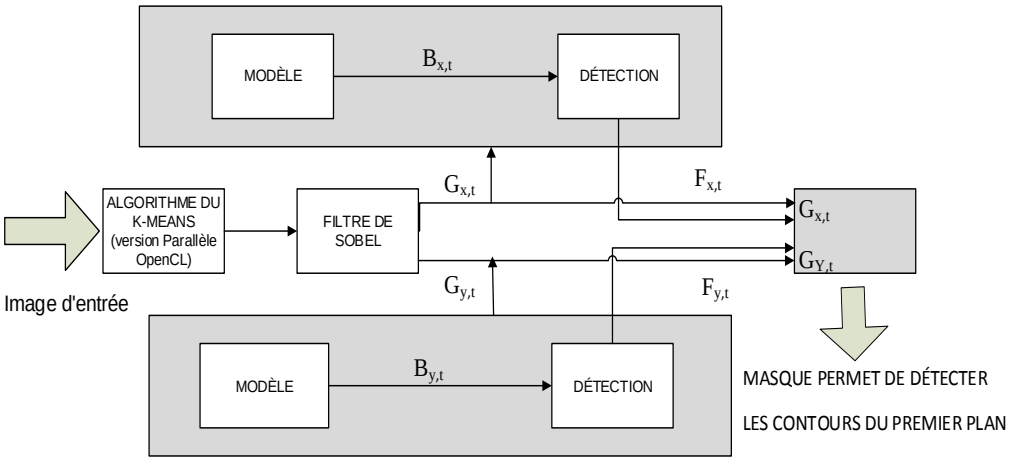


Figure 5-7: Diagramme de blocs de la méthode de soustraction de l'arrière-plan basée sur les contours (version parallèle OpenCL).

À chaque entrée d'un frame, on applique l'algorithme séquentiel du kmeans ou parallèle en utilisant OpenCL voir les deux figures 5.6 et 5.7 puis la sortie du K-means on applique le filtre de Sobel pour déterminer les contours dans les directions X et Y. Ensuite, nous utilisons la moyenne de chaque direction pour estimer le gradient du modèle de background dans les deux directions. La différence entre le frame réel et le modèle d'arrière-plan estimé est utilisée pour la mise à jour, voir les équations 5.1 et 5.2. Deux masques du foregroundsont obtenus, l'un pour la direction X et l'autre pour la direction Y, en comparant le gradient du frame d'entrée avec le modèle d'arrière-plan pour la direction correspondante (voir l'équation 5.3).

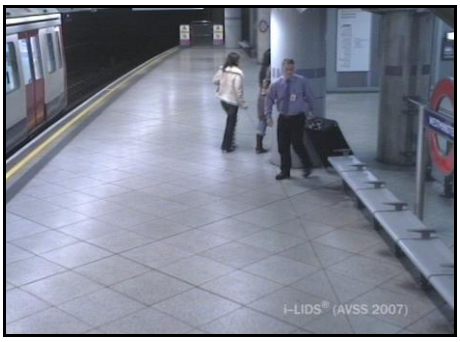
$$B_{x,t}(x, y) = B_{x,t-1}(x, y) + \alpha D_{x,t}(x, y) \quad (5.1)$$

$$D_{x,t}(x, y) = G_{x,t} - B_{x,t}(x, y) \quad (5.2)$$

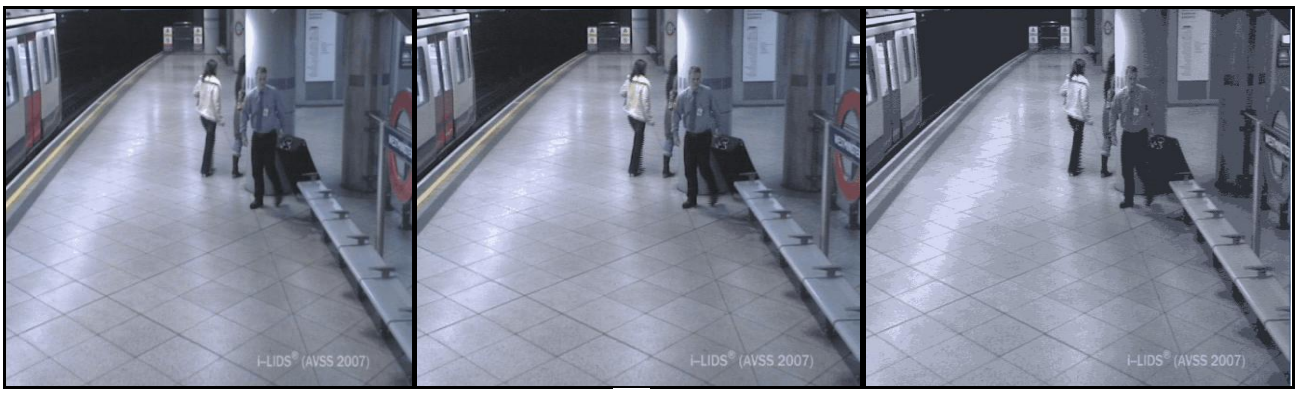
$$F_{x,t}(x, y) = hyst(|D_{x;t}|, T_{low}, T_{high}) \quad (5.3)$$

Avec $B_{x,t}$, $D_{x,t}$, $F_{x,t}$, $G_{x,t}$ et α sont respectivement le modèle d'arrière-plan dans la direction X, la différence entre le modèle d'arrière-plan et l'image de gradient actuel, le masque binaire des contours du premier plan, la différence de gradient dans la direction X, et le taux d'apprentissage (la même chose est valide pour la direction Y).

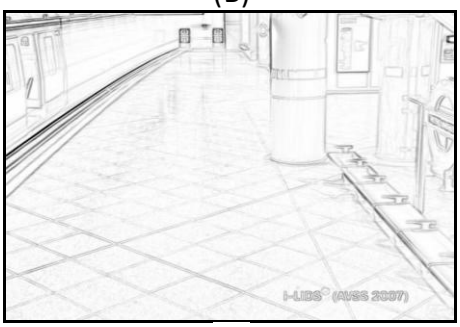
$F_{x,t}$ et $F_{y,t}$ sont calculés en utilisant une fonction de seuillage. Les valeurs du pixel supérieures à T_{high} sont mises à 1, tandis que celles inférieures à T_{low} sont mises à 0. Cependant, les valeurs de pixel entre T_{low} et T_{high} sont mises à 1 si l'un des pixels des huit voisins connectés a une valeur supérieure à T_{high}; sinon, elles sont mises à 0. Enfin, le masque des contours de premier plan F est obtenu en utilisant un opérateur OU entre $F_{x,t}$ et $F_{y,t}$. Contrairement aux auteurs dans [39] qui utilisent un modèle à court terme avec un taux d'apprentissage élevé combiné à un modèle à long terme avec un faible taux d'apprentissage pour supprimer les contours en mouvement bruités et indésirables, dans ce travail, et pour des raisons de simplicité, nous utilisons un seul modèle pour détecter les contours en mouvement. La figure 5.8 montre le modèle d'arrière-plan construit et le masque de sortie des contours de premier plan.



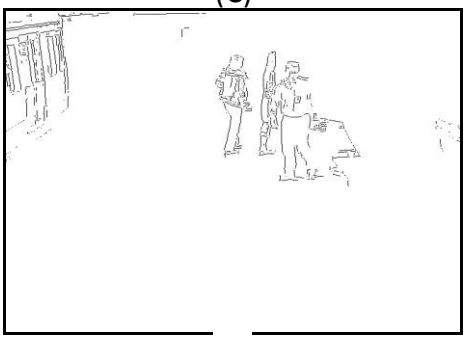
(A)



(B)



(C)



(D)

Figure 5-8: Détection des contours en mouvement A). Frame d'entrée B) appliquer k-means clustering parallèle aux images avec k=80, k=50 et k=10. C). frame d'arrière-plan D). Frame du premier plan.

5.3.2 **Détection des contours stables**

Comme indiqué ci-dessus, les contours stables sont extraits en utilisant une accumulation temporelle sur le masque des contours en mouvement. A chaque entrée de frame, un pixel du masque d'accumulation temporelle est incrémenté de la valeur 2 si le pixel situé au même endroit dans le masque des contours en mouvement a la valeur 1, voir l'équation 5.4.

$$ACCt(x, y) = \begin{cases} ACCt-1(x,y)+1 & \text{if } (Ft(x,y) = 1 \ \&i\%\Delta = 0) \\ ACCt-1(x,y)-1 & \text{if } (Ft(x,y) = 0 \ \&i\%\Delta = 0) \end{cases} \quad (5.4)$$

Pour éviter de prendre en compte les objets temporairement statiques et ceux qui se déplacent lentement, ACC est incrémenté toutes les Δ images au lieu de faire une accumulation image par image.

Des expériences ont été réalisées pour identifier la bonne valeur de Δ , lorsque la valeur de Δ est fixée à 10, le masque ne capture que les informations pertinentes, c'est-à-dire les pixels des contours réels des objets statiques.

Extraction de contours stables

Le masque de sortie SEMask contient les contours stables résultants et AO_{time} est le seuil permettant de dire si un contour appartient à un objet stable, voir l'équation 5.5.

$$SEMask_t(x, y) = hyst\ (ACC_t(x, y), AO_{time}/2, AO_{time}) \quad (5.5)$$

Enfin, les magnitudes des contours stables sont extraites en effectuant une opération ET entre les masques G_x et G_y, puis en appliquant une suppression non maximale (NMS)

Pour obtenir les contours nécessaires à la classification en utilisant les formules suivantes :

$$Sg_{x,t}(x, y) = \begin{cases} Gx,t(x,y) & SEMask(x,y) = 1 \\ 0, & \text{otherwise} \end{cases}$$

$$Sg_{y,t}(x, y) = \begin{cases} Gy,t(x,y) & SEMask(x,y) = 1 \\ 0, & \text{otherwise} \end{cases} \quad (5.6)$$

5.3.3 **La classification**

Lorsqu'un nouvel objet est formé à partir du SEMask, il faut vérifier s'il s'agit d'un véritable objet abandonné ou une fausse détection. Les objets abandonnés n'ont pas une forme régulière, donc il n'est pas possible d'appliquer une détection d'objet basée sur l'apprentissage pour vérifier la nature de l'objet candidat. Par conséquent, un modèle plus général est nécessaire pour classer ces candidats.

Deux scores sont utilisés pour filtrer les fausses détections [1]. Le premier score est le score Objectness, il est utilisé pour vérifier si les contours enfermés dans la boîte de délimitation représentent ou non la limite d'un véritable objet la boîte englobante représentent ou non le contour d'un véritable objet en fonction de l'orientation des contours stables dans leur boîte englobante. Le deuxième score est le score de Staticnessqui est basé sur les contours stables inclus dans le rectangle dans la boîte englobante et sur la connectivité pour vérifier et filtrer les objets avec un mouvement interne qui entraîne ces défauts, comme une personne immobile.

5.3.3.1 **Score Objectness :**

Nous utilisons un score adapté Objectness [1], qui est plus convenable pour le cas des objets de type bagage. Une boîte englobante (BB) d'un candidat AO avec une largeur BB_W et une longueur BB_L, est divisée en quatre régions : gauche, droite, bas et haut, comme le montre la figure 5.9. Ensuite, en utilisant les orientations des contours, le score ***Objectness*** est calculé en mesurant la convexité de l'objet enfermé dans chacune de ces régions. Pour ce faire, nous utilisons les représentations des groupes de contour et leurs directions de gradient moyennes.

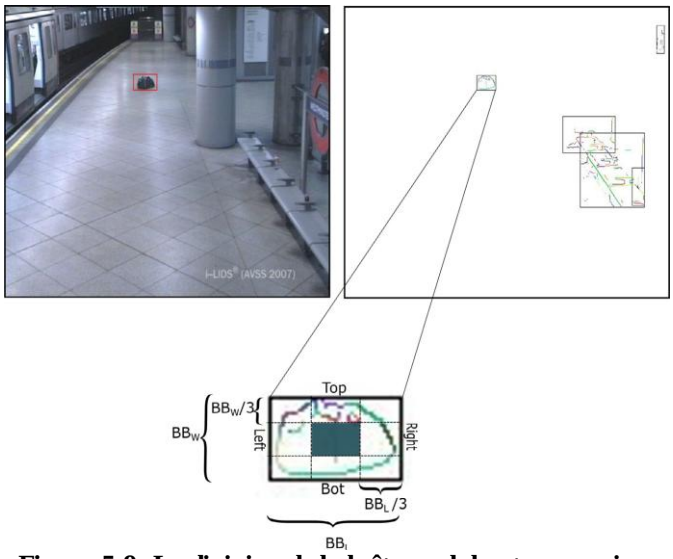


Figure 5-9: La division de la boîte englobante en regions

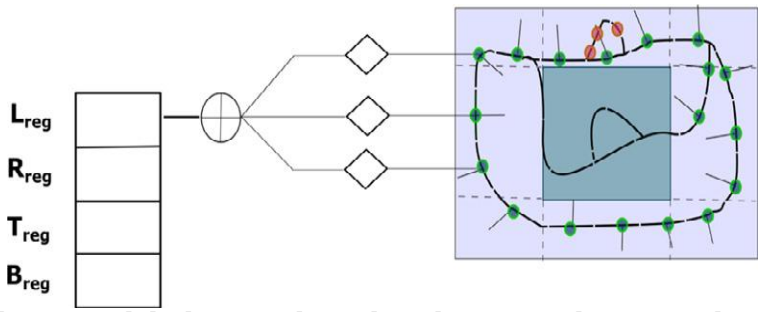


Figure 5-10: . Illustration de la direction du gradient des groupes de contours du candidat AO. Les groupes de contours avec un cercle vert sur la région de gauche satisfont l'équation (5.10).

$T_{reg} = T_{reg} + segLength_i$ if $| \sin(\theta_i)^2 - 1| < \sigma$ (5.7)
$B_{reg} = B_{reg} + segLength_i$ if $| \sin(\theta_i)^2 - 1| < \sigma$ (5.8)
$R_{reg} = R_{reg} + segLength_i$ if $| \sin(\theta_i)^2| < \sigma$ (5.9)
$L_{reg} = L_{reg} + segLength_i$ if $| \sin(\theta_i)^2| < \sigma$ (5.10)

Où θ_i et $segLength_i$ sont respectivement l'orientation moyenne et la longueur du $i^{ème}$ groupe de contour dans chaque région.

T_{reg} , B_{reg} , R_{reg} et L_{reg} sont respectivement les sommes des longueurs des groupes de contours qui satisfassent la condition de convexité dans les régions Haut, Bas, Droit et Gauche (Figure 5.10). Pour les régions Haut et Bas, les longueurs des groupes de contours avec une orientation moyenne proche de $\beta = (\pi/2$ ou $3\pi/2)$ sont accumulées lorsque ($|\sin(\theta i)\ 2 - \sin(\beta)\ 2| < \sigma$) (Eq. (5.7) et (5.8)). Pour les régions droit et gauche, les longueurs des groupes de contours avec une orientation moyenne proche de $\beta = (0$ ou $\pi)$ sont accumulées lorsque ($|\sin(\theta i)\ 2 - \sin(\beta)\ 2| < \sigma$). (Eq. (5.9) et (5.10)). Enfin, nous comparons T reg et B reg avec la longueur BB L de la boîte englobante, et nous comparons R reg et L reg avec la hauteur BB W de la boîte englobante, en définissant le score S_b suivant :

$$Sb = \frac{\lambda}{((Lreg - BBW) * (Rreg - BBW) * (Treg - BBL) * (Breg - BBL))2} \quad (5.11)$$

Ou λ est une constante.

5.3.3.2 **Score Staticness :**

Les fausses alarmes peuvent être générées par des personnes immobiles ou des objets ayant des mouvements faibles et internes. Pour résoudre ce problème, nous proposons un score permettant de vérifier la stabilité des limites de l'objet dans le temps. Nous calculons un score Staticness en mesurant la stabilité des contours de l'objet. Les contours sont très sensibles au mouvement et il est donc facile de capturer tout mouvement petit et discret. Formellement, les boîtes englobantes dans le SEMask, contenant des contours stables très fragmentés, sont peu susceptibles d'être des objets abandonnés. Nous calculons un score Cb en mesurant la

connectivité des contours de l'objet en vérifiant les connexions de chaque groupe de contours avec ses voisins (Eq. (5.12)).

Nous partons de l'hypothèse que pour un objet avec une limite simple, chaque groupe de contour a au moins deux connexions avec ses voisins, et donc les groupes de contour avec moins de deux connexions pénaliseront le score. Le score aide à filtrer les fausses alarmes résultant d'objets non mobiles, mais avec des mouvements internes comme les personnes immobiles.

$$C_b = \prod \frac{|\varphi(i)|}{2} \quad (5.12)$$

$\varphi(i)$ est l'ensemble des connexions inter-contour pour le $i^{ième}$ groupe de contours.

Nous utilisons deux seuils T 1 et T 2 pour les deux scores S_b et C_b respectivement. Si S_b est supérieur à T1 et C_b est supérieur à T2, alors l'objet intérieur est validé et considéré comme un véritable objet abandonné. Pour éviter de prendre en compte les contours d'objets mobiles qui occultent le candidat AO, seules les distributions de contour ayant une valeur d'accumulation supérieure à (AO_{time} /2) sont considérées dans le score.

5.4 **Les Résultats expérimentaux**

Pour les expériences, nous utilisons un ordinateur avec un processeur I3 8350k @4.0 GHz. Nous utilisons le langage de programmation C++ pour répondre aux exigences en temps réel de l'application de surveillance visuelle. La bibliothèque de vision informatique OpenCV est utilisée pour l'acquisition et le prétraitement. L'implémentation de l'algorithme parallèle proposé est évaluée par rapport à l'algorithme K-means version séquentielle, pour cela en a utilisé un GPU Nvidia GeForce GTX 1060 ("Pascal") avec 6 GB de mémoire et une mémoire locale de 49152(48KB). Le langage de programmation parallèle est OpenCL 1.2 pour le code GPU.

Nous exécutons k-means en parallèle le but de cette étape était de réduire le temps d'exécution du k-means qui a été utilisé pour générer l'arrière-plan qui doit être utilisé dans le traitement suivant : détection de premier plan et actualisation de l'arrière-plan.

Pour comparer la version séquentielle du K-means avec la version parallèle proposée dans ce chapitre, le temps d'exécution du clustering est mesuré dans différents ensembles de données

La méthode est évaluée sur les ensembles de données accessibles au public utilisés dans la littérature sur la détection des objets abandonnés : I-LIDS'sAVSS2007 [40], PETS2006 [41], PETS2007 [42], et ABODA [43].

Paramètres & seuils

Le tableau 5.1 résume les valeurs des paramètres utilisés dans l'algorithme. $\mathbf{T_{low}}$ = 40,$\mathbf{T_{high}}$ = 70, et $\boldsymbol{\alpha}$= 0:005 sont pour le traitement de background. Ils ont été choisis de manière à ce que le temps nécessaire à l'absorption d'un objet dans le modèle de background soit supérieur à la durée d'abandon $\mathbf{AO_{time}}$ où$\mathbf{AO_{time}}$ est un paramètre défini par l'utilisateur. σ est le seuil pour qu'un groupe de contours soit pris en compte dans le score a été fixé à 0,5, ce qui signifie que la différence d'orientations entre un groupe de contours et la limite de sa boîte de délimitation relative doit être inférieure à π /4. $\boldsymbol{\lambda}$ est une constante et est fixée à 10^8. Les seuils T1 et T2, pour S_b et C_b respectivement.

Tableau 5-1: Résumé des paramètres utilisés dans l'algorithme

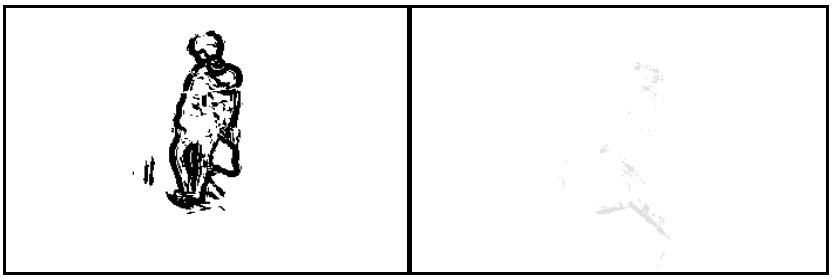
Paramètre	**α**	T_{low}	T_{high}	**σ**	T1	T2	**λ**
Valeur	0.005	40	**70**	**0.5**	10^{-5}	10^{-5}	**Constante**

✓ Les tailles des images sur lesquelles nous avons testé notre algorithme sont 320×240 et 720×480.

✓ Le nombre de clusters est varié entre {2, 3, 4, 16, 32, 64, 128, 256, 512,1024}.

✓ Le nombre d'itérations est varié entre {5,10,15,30,40,50,60,70,80,90,100}

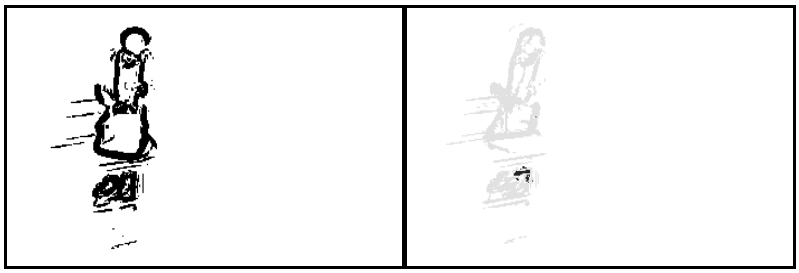
(A) (B)



(C) (D)

Figure 5-11: A: résultat de la méthode proposée. B: application de l'algorithme du k-means (version parallèle) aux images d'entrées. C: Détection des contours en mouvement. D: Détection des contours stable

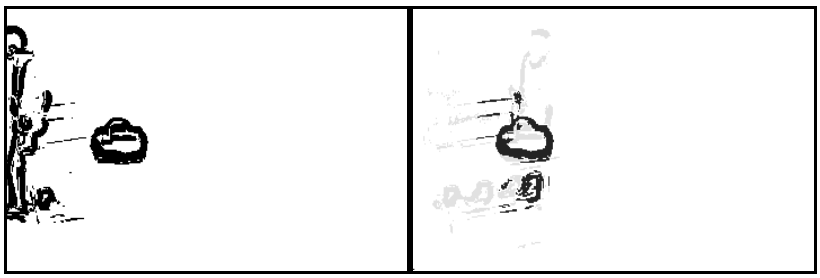
(A) (B)



(C) (D)

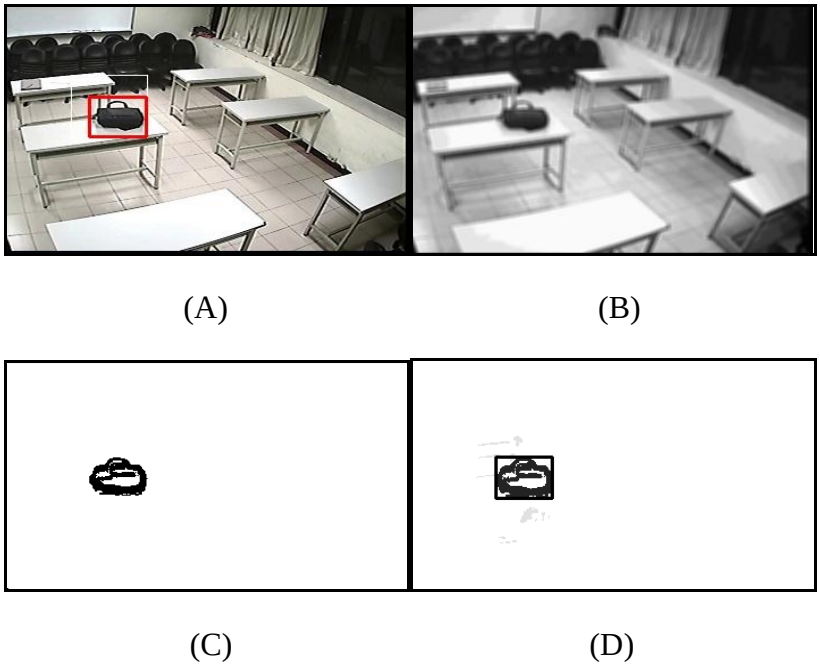
Figure 5-12: A: résultat de la méthode proposée. B: application de l'algorithme du k-means (version parallèle) aux images d'entrées. C: Détection des contours (version parallèle) aux images d'entrées. C: Détection des contours en mouvement. D: Détection des contours stables.

(A) (B)



(C) (D)

Figure 5-13: A: résultat de la méthode proposée. B: application de l'algorithme du k-means (version parallèle) aux images d'entrées. C: Détection des contours en mouvement. D: Détection des contours stables.



(A) (B)

(C) (D)

Figure 5-14: A: résultat de la méthode proposée. B: application de l'algorithme du k-means (version parallèle) aux images d'entrées. C: Détection des contours en mouvement. D: Détection des contours stables

Les figures 5.11, 5.12, 5.13, 5.14 montrent les résultats de notre méthode. La méthode proposée détecte tous les objets abandonnés avec précision tout en filtrant les fausses alarmes. Dans ce scénario, notre module de classification filtre parfaitement les fausses alarmes, contrairement aux d'autres méthodes où fausses détections résultantes des ordures en mouvement sont classées comme de véritables objets abandonnés.

Les figures 5.15, 5.16, 5.17, 5.18 expliquent les résultats on compare le temps de l'exécution du k-means séquentielle et k-means parallèle qui utilise la plateforme OpenCL qui a été fait sur plusieurs cas en variant la taille des images, le nombre d'itérations et le nombre de clusters. Les résultats montrent que l'accélération de l'implémentation parallèle de l'algorithme du k-means peut attend de 9X par apports a la version séquentielle du k-means pour une taille d'image de 320 × 240 et l'accélération maximale de 7X par apports a la version séquentielle du k-means pour une taille d'image de 720 × 480 en gardant la même précision.

Nous calculons le temps d'exécutions de chaque algorithme (algorithme 5.1 et 5.2). Puisque la complexité de calcul de la recherche du centroïde le plus proche est O(nkd+nk), qui est beaucoup plus grande que O(nd) du calcul du nouveau centroïde, une approximation raisonnable du nombre total d'opérations peut être obtenue par :

$$OP = n \times k \times (d + d + d - 1) \times iter \quad (5.13)$$

Pour chaque point de données, le premier d de l'équation (5.13) est le nombre de soustractions, le second d est le nombre de multiplications, le troisième terme (d – 1) est le nombre d'additions et iter est le nombre d'itérations.

Le temps de calcul augmente également avec l'augmentation du nombre d'itérations, le nombre de clusters et la taille d'image, comme le montre les figures 5.15, 5.16, 5.17, 5.18 .En outre, nous pouvons également observer que le temps consommé pour "calculer de nouveaux centroïdes" change légèrement lorsque les paramètres changent, en raison de la faible complexité de calcul et de notre algorithme 5.2 ; le tableau 5.2.

Tableau 5-2: Distribution du temps d'exécution de l'algorithme proposé K-means, en millisecondes

	ensemble de données	Déterminer le centroïde le plus proche	Calcul de nouveaux centroïdes
CPU Algorithme5.1	Video1 240x320, k= 16 , iter=100	2326	250
GPU Algorithme5.2	Video1 240x320, k= 16 , iter=100	285	20

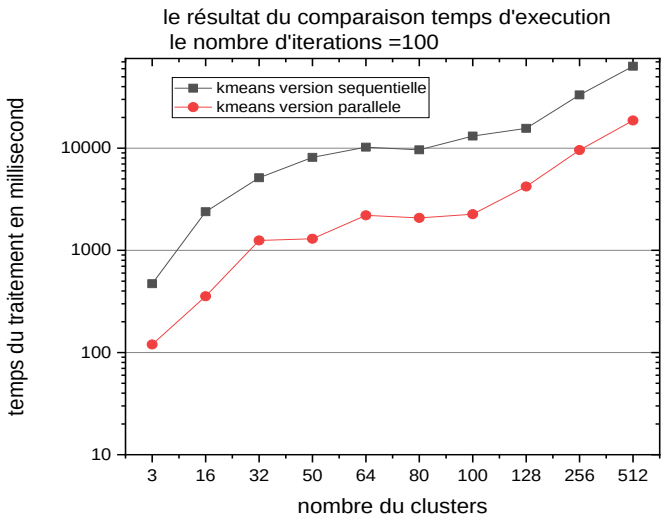


Figure 5-15: Une comparaison entre k-means la version séquentielle et la version parallèle ou on a varié le nombre de clusters (le nombre d'itérations est fixé à 100 et résolution 240x320).

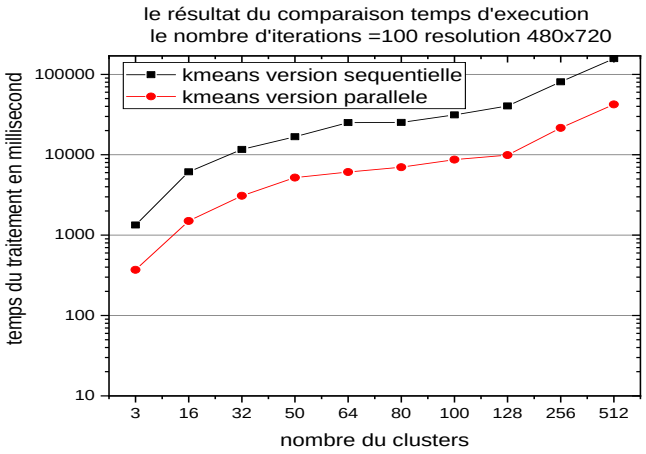


Figure 5-16: Une comparaison entre k-means la version séquentielle et la version parallèle ou on a varié le nombre de clusters (le nombre d'itérations est fixé à 100 et résolution 480x720).

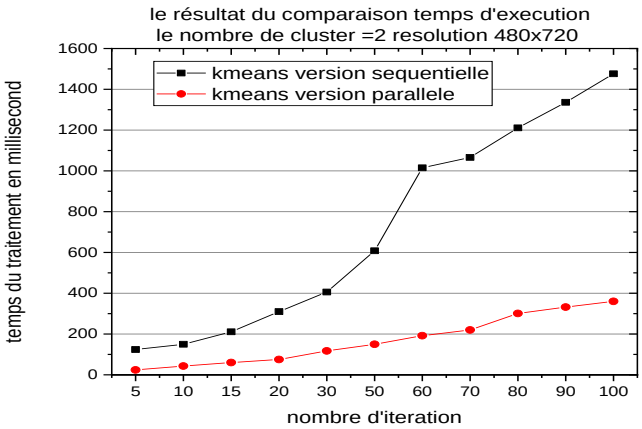


Figure 5-17: Une comparaison entre k-means la version séquentielle et la version parallèle ou on a varié le nombre d'itérations (le nombre de clusters est fixé à 2 et résolution 480x720).

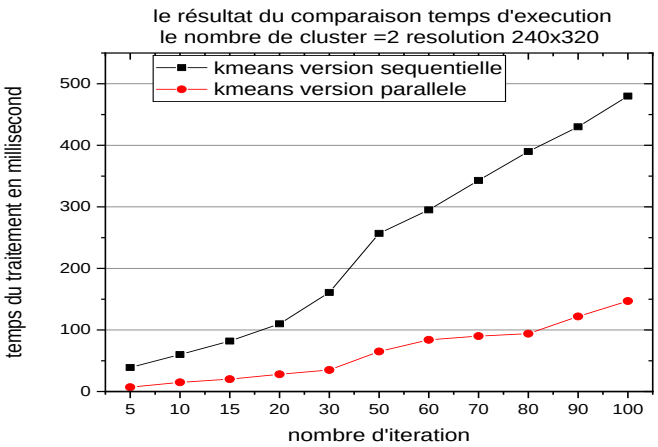


Figure 5-18: Une comparaison entre k-means la version séquentielle et la version parallèle ou on a (en prenant des différentes valeurs du nombre d'itération) varié le nombre d'itérations (le nombre de clusters est fixé à 2 et résolution 240x320).

Tableau 5-3: Les résultats de la comparaison entre PETS2006 et AVSS2007

La méthode	PETS2006			AVSS2007		
	R (Recall)	P (Precision)	FM (F-measure)	R (Recall)	P (Precision)	FM (F-measure)
L'approche proposée	1.0	1.0	1.0	1.0	1.0	1.0

La rapidité globale de calcul est de **18fps** en CPUet**41fps** en GPU Où (**FPS**) se réfère à **F**rame **P**er **S**econd, pour une résolution de 240x320 et le nombre d'itération égale à 5 et le nombre du cluster égal à 3.

Comparaison du travail proposé avec la littérature

Dans cette section, nous comparons nos meilleures performances obtenues avec d'autres meilleures performances de l'état de l'art, Comme le montre le tableau 5.3, nous obtenons une accélération de plus de X8 par rapport au Séquentiel k-means, tandis que le meilleur résultat publié à ce jour est obtenu par [24] qui utilise Pthread et obtient une accélération de X1.5 par rapport au Séquentiel k-means.

Tableau 5-4: Comparaison du temps de traitement (k-means parallèle) avec d'autres méthodes de l'état de l'art, la résolution utilisée est 320x240

Méthode	Rapidité du traitement	rapidité globale de calcul
Notre approche	9	41fps
[24]	1.5	10 fps

5.5 Conclusion

Dans ce chapitre, nous avons développé un système de détection automatique des objets abandonnés. Nous avons également proposé un algorithme k-means parallèle en utilisant la plate forme OpenCL, où l'amélioration de l'algorithme provient de la réduction considérable du temps d'exécution du k-means traditionnel. Des expériences ont été réalisées sur de nombreux cas de vidéos provenant d'un ensemble de données. Les tests ont montré qu'il y avait un gain de temps d'environ 9X fois par apport à l'algorithme K-means traditionnel, en gardant la même précision. Notre méthode a prouvé sa robustesse face aux problèmes classiques de la vidéosurveillance (changements d'illumination, zones d'encombrement) et son

efficacité dans des scénarios réels. Notre système a été conçu pour traiter des vidéos en temps réel pour la détection des objets abandonnés.

Dans la conclusion, nous présentons un résumé de la thèse et nous donnons des perspectives pour des futurs travaux connexes.

5.6 **Références**

[1] I. Dahi, M. Chikr el Mezouar, N. Taleb, and M. Elbahri, "An Edge-based Method for Effective AbandonedLuggageDetection in Complex Surveillance Videos," Computer Vision and Image Understanding, vol. 158, no. C, pp. 141–151, 2017.
[2] M. Bhargava, C.-C. Chen, M. S. Ryoo, and J. K. Aggarwal, "Detection of abandonedobjects in crowdedenvironments," in Proceedings of AVSS, 2007, pp. 271–276.
[3] H.-H. Liao, J.-Y. Chang, and L.-G. Chen, "A LocalizedApproach to AbandonedLuggageDetectionwithForeground-MaskSampling," in Proceedings of AVSS, 2008, pp. 132–139.
[4] F. Porikli, Y. Ivanov, and T. Haga, "Robustabandonedobjectdetectionusing dual foregrounds," EURASIP Journal on Advances in Signal Processing, vol. 2008, p. 30, 2008.
[5] Y.-L. Tian, R. Feris, and A. Hampapur, "Real-Time Detection of Abandoned and RemovedObjects in ComplexEnvironments," in Proceedings of VS Workshop, 2008.
[6] M. Bhargava, C.-C. Chen, M. S. Ryoo, and J. K. Aggarwal, "Detection of objectabandonmentusing temporal logic," Machine Vision and Applications, vol. 20, no. 5, pp. 271–281, 2009.
[7] J. Wen, H. Gong, X. Zhang, and W. Hu, "Generative model for abandonedobjectdetection," in Proceedings of ICIP, 2009, pp. 853–856.
[8] J.-Y. Chang, H.-H. Liao, and L.-G. Chen, "LocalizedDetection of AbandonedLuggage," EURASIP Journal on Advances in Signal Processing, vol. 2010, no. 1, 2010.
[9] P. Forczma´nski and M. Seweryn, "Surveillance Video Stream AnalysisUsing Adaptive Background Model and Object Recognition," in Proceedings of ICCVG, 2010, pp. 114–121.
[10] S. Kwak, G. Bae, and H. Byun, "Abandonedluggagedetectionusing a finite state automaton in surveillance video," Optical Engineering, vol. 49, no. 2, pp. 027 007–1–027 007–10, 2010.
[11] G. Szwoch, P. Dalka, and A. Czy˙zewski, A Framework for AutomaticDetection of AbandonedLuggage in Airport Terminal. Springer, 2010, pp. 13–22.
[12] Y. Tian, R. S. Feris, H. Liu, A. Hampapur, and M. T. Sun, "RobustDetection of Abandoned and RemovedObjects in Complex Surveillance Videos," IEEE Transactions on Systems, Man, and Cybernetics, vol. 41, no. 5, pp. 565–576, 2011.
[13] K. Lin, S. Chen, C. Chen, D. Lin, and Y. Hung, "Left-LuggageDetectionfromFinite-State-Machine Analysis in Static-Camera Videos," in Proceedings of ICPR, 2014, pp. 4600–4605.
[14] K. Lin, S. C. Chen, C. S. Chen, D. T. Lin, and Y. P. Hung, "Abandoned Object Detection via Temporal ConsistencyModeling and Back- TracingVerification for Visual Surveillance," IEEE Transactions on Information Forensics and Security, vol. 10, no. 7, pp. 1359–1370, 2015.
[15] N. T. Pham, K. Leman, J. Zhang, and I. Pek, "Two-stage unattendedobjectdetectionmethodwithproposals," in Proceedings of ICSIP, 2017, pp. 1–4.
[16] K. C. Smith, P. Quelhas, and D. Gatica-Perez, "DetectingAbandonedLuggage Items in a Public Space," in Proceedings of PETS Workshop, 2006.

[17] G. Szwoch, "Extraction of stable foreground image regions for unattendedluggagedetection," Multimedia Tools and Applications, vol. 75, no. 2, pp. 761–786, 2016.
[18] J. Ferryman, D. Hogg, J. Sochman, A. Behera, J. A. Rodriguez- Serrano, S. Worgan, L. Li, V. Leung, M. Evans, P. Cornic et al., "Robustabandonedobjectdetectionintegratingwide area visual surveillance and social context," Pattern Recognition Letters, vol. 34, no. 7, pp. 789–798, 2013.
[19] H. Kong, J.-Y. Audibert, and J. Ponce, "Detectingabandonedobjectswith a moving camera," IEEE Transactions on Image Processing, vol. 19, no. 8, pp. 2201–2210, 2010.
[20]**.** Bouwmans, T., Porikli,F. , Höferlin , B. and acavant**,**A. V. 2015. *Background Modeling and ForegroundDetection for Video Surveillance***.** CRC Press. E-book.
[21]**.** Butler, D., Bove, V. and Shridharan, S. **2005**. Real time adaptive foreground/background segmentation. *EURASIP*, **14**: 2292–2304.

[22]**.** Duan, X., Sun, G. and Yang,T. **2011**. Movingtargetdetectionbased on genetic k-meansalgorithm .*International Conference on Communication TechnologyProceedings, ICCT*. 978-1-61284-307-0/11/$26.00 ©2011 IEEE: 819-822.
[23]**.** Sinha, S. and Mareboyana,M. **2014**. Video Segmentation into Background and ForegroundUsingSimplifiedMean Shift Filter and K-MeansClustering .ASEE . University of Bridgeport

[24]Yossra Hussain Ali, Mohammed Rajah Mohammed .Background modeling in video surveillance by usingparallelcomputing. Iraqi Journal of Science, 2017, Vol. 58, No. 3B, pp: 1516-1522. DOI: 10.24996/ ijs.2017.58.3B.16

[25]Daoudi, S., Zouaoui, C.M.A., El-Mezouar, M.C.,Taleb,N. (2019). A comparative study of parallel CPU/GPU implementations of the K-MeansAlgorithm. In 2019 International Conference on Advanced Electrical Engineering (ICAEE), pp. 1-5.https://doi.org/10.1109/ICAEE47123.2019.9014783

[26] Daoudi, S., Zouaoui, C.M.A., El-Mezouar, M.C., Taleb, N. (2021). Parallelization of the K-means++ clusteringalgorithm. Ingénierie des Systèmes d'Information, Vol. 26, No. 1, pp. 59-66. https://doi.org/10.18280/isi.260106.

[27] Li, Y., Zhao, K., Chu, X., Liu, J.: Speeding up K-MeansAlgorithm by GPUs. Journal of Computer and System Sciences v79(2), 216–229 (2013).

[28] Fuhui Wu, Qingbo Wu, Yusong Tan, Lifeng Wei, LisongShao, and Long Gao. 2013. A vectorized k-meansalgorithm for intelmanyintegratedcore architecture. In International Workshop on Advanced ParallelProcessing Technologies. Springer, 277–294.

[29] Sharafeddin M, Partamian H, Awad M, Saghir M.A, Akkary H, Artail H, Hajj H, Baydoun M .Towardsdistributedacceleration of image processing applications using

reconfigurable active SSD clusters: a case study of seismic data analysis. Int J High Perform Comput Networking (in press). (2016).

[30] Tang, Q. Y., Khalid, M. A. : Accelerationof K-MeansAlgorithmUsingAltera SDK forOpenCL. ACM TransactionsonReconfigurableTechnologyandSystems(TRETS), Vol. 10, No. 1, 6. (2016).

[31]Baramkar, P.P.; Kulkarni, D.B. Review for K-Means On Graphics ProcessingUnits (GPU). Int. J. Eng.Res. Technol. **2014**, 3, 1911–1914.

[32]. Zechner, M.; Granitzer, M. K-Means on the Graphics Processor: Design and ExperimentalAnalysis. Int. J.Adv. Syst. Meas. **2009**, 2, 224–235. [CrossRef]

[33] Hong-tao, B.; Li-li, H.; Dan-tong, O.; Zhan-shan, L.; He, L. K-Means on CommodityGPUswithCUDA.InProceedings of the WRI World Congress on Computer Science and Information Engineering, Los Angeles,CA, USA, 31 March–2 April 2009; pp. 651–655. [CrossRef]

[34]Fakhi, H.; Bouattane, O.; Youssfi, M.; Hassan, O. New optimized GPU version of the k-meansalgorithmfor large-sized image segmentation. In Proceedings of the Intelligent Systems and Computer Vision, Fez,Morocco, 17–19 April 2017; pp. 1–6. [CrossRef]

[35]. Baydoun, M.; Dawi, M.; Ghaziri, H. Enhancedparallelimplementation of the K-Meansclusteringalgorithm.InProceedings of the 3rd International Conference on Advances in Computational Tools for EngineeringApplications (ACTEA), Beirut, Lebanon, 13–15 July 2016; pp. 7–11. [CrossRef].

[36]. Lutz, C.; Bress, S.; Rabl, T.; Zeuch, S.; Markl, V. Efficient k-means on GPUs. In Proceedings of the 14thInternationalWorkshop on Data Management on New Hardware, Huston, ID, USA, 11 June 2018. [CrossRef]

[37].MacQueen,J.B.(1967).SomeMethodsforclassification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on MathematicalStatistics andProbability.Berkeley,UniversityofCaliforniaPress, 1:281-297.https://www.bibsonomy.org/bibtex/25dcdb8cd9fba78e0e791af619d61d66d/enitsirhc.

[38]. Jipkate, B. R. and Gohokar,V. **2012**. A Comparative Analysis of Fuzzy C-MeansClustering and K MeansClusteringAlgorithms .*IJCER*, **2**(3): 737-739 .

[39] SebastianGruenwedel, Peter Van Hese, andWilfried Philips. “An edgebased

approach for robustforegrounddetection". In: International Conference on Advanced Concepts for Intelligent Vision Systems. Springer. 2011, pp. 554–565.

[40]. "AVSS2007". URL: http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html.
[41]. "PETS2006". URL: http://www.cvg.reading.ac.uk/PETS2006/data.html.
[42]. "PETS2007". URL: http://www.cvg.reading.ac.uk/PETS2007/data.html.
[43]. "ABODA". URL: http : / / imp . iis .sinica .edu .tw / ABODA /index.html.

Conclusion générale

Bilan

Nous avons développé un système de détection automatique des objets abandonnés. La nouveauté de notre méthode est l'utilisation de l'algorithme du clustering pour obtenir le background initial. Cette thèse vise pour son essentiel la conception et la mise en œuvre d'algorithmes parallèles temps réels de clustering réputés d'être très chronophages et énergivores dans la détection des objets abandonnés.

Les implémentations parallèles des algorithmes du clustering K-means et son extension K-means++ en utilisant la plateforme Open Computing Language (OpenCL) synthétisée pour GPU ont obtenu de bons résultats par rapport aux CPU. Cependant, elles n'ont pas pu rivaliser avec les GPU que pour certaines tailles de problèmes. Dans le cas de traitement de petits ensembles de données, l'implémentation parallèle du K-means en utilisant OpenMP a surpassé de manière significative l'implémentation optimisée du GPU. Cependant, l'implémentation parallèle de l'algorithme K-means++ en utilisant OpenCL sur GPU et la technologie Streaming SIMD Extension (SSE) surpasse de manière significative l'implémentation séquentielle du K-means++. Il est important de noter que l'architecture du CPU et du GPU a été optimisée pour gérer très bien ce types d'algorithmes.

L'algorithme K-means parallèle en utilisant la plateforme OpenCL a été implémenté dans le système de détection automatique des objets abandonnés. Les résultats montrent qu'il y a un gain de temps qui surpasse de manière significative l'algorithme K-means traditionnel, en gardant la même précision pour la détection les objets abandonnés. Notre système a été conçu pour traiter des vidéos en temps réel pour la détection des objets abandonnés.

Perspectives

Comme extensions futures, on vise l'implémentation des algorithmes développés sur des accélérateurs à faible coût et à consommation énergétique modérée, tels que les FPGA, à l'aide du langage OpenCL en introduisant la méthodologie de conception de systèmes embarqués le Codesign, qui va répondre à tous les objectifs d'un système temps réel, en faisant une comparaison entre les résultats obtenus précédemment pour le GPU.

Il serait intéressant d'implémenter d'autres algorithmes parallèles tels que K-nearest neighbors en utilisant AOCL « Altera SDK for OpenCL » sur FPGA, pour déterminer si le FPGA pourrait atteindre des performances comparables ou supérieures à celles du CPU et GPU.

Altera Corporation a récemment lancé la nouvelle génération de FPGA 10. Les FPGA Arria 10 [1] haut de gamme et Stratix 10 [2] haut de gamme seront pris en charge par AOCL « Altera SDK for OpenCL ». La nouvelle génération de FPGA utilise des unités DSP à virgule flottante pour mettre en œuvre les fonctions à virgule flottante. Il serait intéressant de compiler les noyaux développés dans cette recherche pour cibler Arria 10 et Stratix 10 et déterminer le gain de performance qui pourrait être obtenue.

La conception de notre système de détection automatique des objets abandonnés par l'algorithme K-means parallèle en introduisant la conception conjointe (Codesign), nécessite des cartes de développements spécifiques telsque Zed Board, ZynqBerry, qui sont dotées d'une architecture qui est le Zynq-SOC. Cette puce fournit les deux ressources nécessaires pour la conception des systèmes embarqués CPU et FPGA.

Références

[1] Altera.com, "Arria 10 - Overview," 2020. [Online]. Available : https://www.altera.com/products/fpga/arria-series/arria-10/overview.html.
[2] Altera.com, "Stratix 10 - Overview," 2020. [Online]. Available : https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html.