
PRÉ-RAPPORT

de stage de master 2

Mention Informatique

DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité: Systèmes Et Applications Répartis

par

Racha Ahmad

Titre

**Exécution parallèle sur cartes graphiques d'un
système de gestion de flux de travail de
calculs numériques**

Responsables de stage :

M. Emmanuel Chailloux

M. Mathias Bourgoïn

Année universitaire 2015-2016

Introduction

Ce pré-rapport décrit mon stage actuellement effectué au sein de l'équipe APR (Algorithmes, Programmes et Résolution) au laboratoire LIP6 à l'Université Pierre-Et-Marie-Curie sous la direction de M. Emmanuel Chailloux et M. Mathias Bourgoïn. Le stage de fin d'étude s'inscrit dans le cadre du Master d'informatique spécialité Systèmes et Applications Répartis (SAR). Il se déroule du 2 février au 31 Juillet 2016.

L'intitulé de mon stage est « Exécution parallèle sur cartes graphiques d'un système de gestion de flux de travail de calculs numériques », Ce projet est en collaboration avec les universités brésiliennes UFRJ¹ et UFF².

1. UFRJ - l'Université Fédérale de Rio de Janeiro

2. UFF - l'Université fédérale Fluminense.

Chapitre 2

État de l'art

Ce chapitre présente le domaine des systèmes de gestion de flux de travail scientifique, l'approche algébrique adoptée par ces systèmes, et les défis rencontrés dans la parallélisation de ces flux. Il introduit également la programmation GPGPU comme une solution intéressante dans le parallélisme des flux de travail scientifiques.

2.1 Les systèmes de gestion de flux de travail scientifique

Le progrès massif dans les différents domaines scientifiques requiert des analyses plus complexes et raffinées sur les expériences scientifiques. Ces expériences à grande échelle sont généralement composées de plusieurs modèles de calcul mathématiques et informatiques. Ils peuvent impliquer l'exécution des simulations sur plusieurs pas de temps, le *fine-tuning* sur un grand ensemble de paramètres. Afin d'assurer une bonne gestion de ces expériences il faut garantir leurs efficacité, cohérence, et reproductibilité [4]. C'est une tâche très importante mais aussi compliquée compte tenu de la nécessité d'indiquer la provenance des données. Dans le but d'accorder une approche systématique pour modéliser et exécuter de telles expériences, de nombreux domaines scientifiques (comme les services d'informations, et l'analyse de signal) utilisent les systèmes de gestion de flux de travail scientifique (SWfMS) [4]. Les SWfMS sont des outils pour modéliser et orchestrer l'exécution des flux de travail scientifique. Ils permettent aux utilisateurs de spécifier un flux de travail composé par des activités (programmes informatiques) et par le flux de données entre ces activités. Cela permet d'enchaîner et de contrôler des différentes tâches pour réaliser un traitement complexe [11].

Dans le cadre de ce stage on s'appuie sur Chiron comme un système de gestion de flux de travail. Chiron (voir Annexe A) a été créée à l'institut COPPE¹ de l'Université Fédérale de Rio de Janeiro (UFRJ) avec l'équipe de recherche de Marta Matosso au laboratoire NACAD.

1. COPPE – Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia.

2.2 La provenance des données dans les systèmes de gestion du flux de travail scientifique

L'un des principaux avantages des systèmes de gestion du flux de travail provient de l'enregistrement de la provenance des données. On définit la provenance des données comme des informations qui permettent de déterminer l'historique de dérivation d'un produit de données, à partir de sa source d'origine [14]. Les deux caractéristiques importantes de la provenance d'un produit de données sont d'une part le produit ancêtre de données à partir de lequel ce produit de données a évolué, et d'autre part le processus de transformation de ce produit ancêtre de données, éventuellement par le biais du flux de travail, qui a aidé à obtenir ce produit de données. Les informations de provenance peuvent être collectées par un modèle orienté processus, où les processus sont les principales entités pour lesquelles la provenance est recueillie, et la provenance des données est déterminée par l'inspection d'entrée et de sortie des produits de données de ces processus. Si les informations de provenance sont disponibles, ils donnent la possibilité aux utilisateurs d'analyser les résultats de flux de travail, et de décider la dépendance et l'ordonnancement entre les activités de ce flux. Des requêtes soumises sur les données de provenance peuvent alimenter automatiquement les ensembles de données d'entrée pour un flux de travail en temps d'exécution. L'historique de dérivation des ensembles de données décrit comment le flux de travail a été spécifié et exécuté. Il peut être utilisé pour répliquer les données sur un autre site, ou le mettre à jour si un ensemble de données est périmé en raison de modifications apportées à ses ancêtres.

2.3 L'approche algébrique pour l'exécution parallèle de flux de travail

Considérant un flux des tâches avec une dépendance de données entre ces tâches, le modèle d'exécution d'un flux de travail est étroitement couplé à ces spécifications [7]. Ce couplage réduit la possibilité d'améliorer les stratégies d'exécution par le système de gestion et par conséquent les améliorations doivent être codées manuellement. Pour répondre à la question de l'optimisation de l'exécution parallèle d'un flux de travail une approche a été proposée dans [11]. Elle consiste à abstraire la gestion d'un flux de travail scientifique sous une approche algébrique, inspirée par l'algèbre relationnelle pour les bases de données. Cela fournit un modèle uniforme de données qui exprime toutes les données d'une expérience par des relations, chaque combinaison de valeurs des paramètres compose un n-uplet, et les activités de gestion de flux de travail consomment et produisent des n-uplets. Cette abstraction a amélioré la conception et la réutilisation du flux de travail, lorsque les activités sont représentées comme des opérateurs algébriques. Les moteurs de gestion de flux de travail peuvent en outre jouer avec des expressions algébriques équivalentes. Ainsi, les SWfMS identifient la façon dont les

données sont structurées et ce qu'il faut attendre de chaque activité, en termes de production et de consommation des données. De cette façon, il est possible d'effectuer des optimisations algébriques et d'adopter des stratégies de distribution intelligentes.

2.4 Le parallélisme dans les systèmes de gestion du flux de travail scientifique

L'ampleur des données produites dans les nombreux domaines scientifiques augmente à un rythme exponentiel, et le progrès de capacité de stockage, de bande passante du réseau, et de puissance de traitement est souvent dépassé. Outre les progrès algorithmiques, la manière canonique pour faire face à l'augmentation des volumes de données est le parallélisme. Cela se reflète dans le développement de l'exécution parallèle des threads sur des puces simples. Ainsi que le développement des infrastructures qui combinent plusieurs machines à des clusters, des grids et des clouds. Les réalisations des exécution des flux de travail parallèle sont généralement désignés afin de fonctionner dans ces environnement de calcul particulier.

Une grille informatique (en anglais, grid) est une infrastructure virtuelle constituée d'un ensemble de ressources informatiques potentiellement partagées, distribuées, hétérogènes, délocalisées et autonomes. Cette infrastructure est qualifiée de virtuelle car les relations entre les entités qui la composent n'existent pas sur le plan matériel mais d'un point de vue logique. L'idée d'un grid est de relier les ressources informatiques afin de résoudre des problèmes de calcul exigeants sans avoir besoin d'un super-ordinateur. Plusieurs SWfMS, tels que Pegasus [6] ou Condor Dagman [5], ont été développés pour utiliser les grids pour l'exécution parallèle des flux de travail de calcul intensif. L'informatique en nuage (en anglais, cloud computing) est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau. Ces ressources de calcul et de stockage sont louables sur demande et sur Internet. Le rendement réel des machines virtuelles louées varie considérablement en fonction de la configuration du matériel et de l'utilisation des ressources sous-jacentes partagées par d'autres utilisateurs. Dans les environnements de grids et de clouds, le transfert important sur des zones étendues des données et les retards dans l'instanciation de grandes quantités des tâches, conduit à une dégradation de performance. Un cluster de calcul est un ensemble d'ordinateurs étroitement connectés et fonctionnant comme un système unique. Un nombre toujours croissant de cœurs par processeur et processeurs par cluster ont conduit à un fort potentiel de parallélisation. Puisque les ressources de calcul en clusters sont étroitement couplés, la localité de données est moins problématique par rapport à des environnements distribués, tels que les grids et les clouds. Les clusters fournissent un environnement plus homogène en termes de performances du processeur et de la latence / bande passante entre les nœuds de calcul. Cependant, l'évolutivité des clusters est limitée et les coûts initiaux d'investissement sont assez élevés, ce qui est problématique. De toute évidence, l'exécution des flux

de travail scientifique présente des défis différents en fonction de l'infrastructure informatique sous-jacente [3]. Les spécifications pour l'exécution en parallèle d'un flux de travail scientifique sont généralement définies avec un langage de script. Ce langage est traité par un moteur de flux de travail qui génère un plan correspondant pour l'exécution parallèle. Ce plan est une mise en correspondance entre l'architecture parallèle des matériels et la planification des tâches correspondants. Développeurs de flux de travail doivent décider l'ordre, les dépendances et les stratégies de parallélisation. Ces décisions resserrent les possibilités de parallélisation, et peuvent donner à manquer de grandes opportunités d'optimisation. Le but de ce stage est d'expérimenter l'accélération GPU comme une nouvelle approche de parallélisme dans le domaine de gestion de flux de travail scientifique. Actuellement, le seul SWfMS qui est en mesure de tirer parti des accélérateurs tels que GPGPU est Swift [8]. Néanmoins, La définition de flux de travail dans Swift est fortement couplée à la procédure d'exécution du flux de travail. Et la provenance de données dans Swift est basée sur l'extraction des données à partir des fichiers de logs, de ce fait, le scientifique peut explorer la base de données de provenance uniquement lorsque l'exécution est terminée.

2.5 La programmation GPGPU

Les cartes graphiques (GPU) sont des dispositifs performants et spécialisés dotés de nombreuses unités de calcul, dédiés à l'affichage et au traitement 3D. La technologie GPGPU est l'abréviation de *general-purpose computing on graphics processing units*, c'est-à-dire un calcul générique sur un processeur graphique. Cette technologie exploite la puissance de calcul des GPUs pour le traitement massivement parallèle, elle permet d'accélérer les portions de code les plus lourdes en ressources de calcul, en les parallélisant sur de nombreuses unités de calcul, le reste de l'application restant affecté au CPU. La programmation GPGPU offre une accélération très élevée pour un large éventail d'applications scientifiques et commerciales, nettement supérieure à celle offertes par une architecture basée uniquement sur des CPUs [10]. Les GPUs offrent une possibilité d'améliorer les exécutions parallèles de flux de travail scientifiques, souvent exprimé avec un grand nombre des tâches de calcul. Il existe deux outils dédiés à réaliser des calculs généralistes Cuda et OpenCL. Ce sont des outils de très bas niveau d'abstraction, ils demandent de manipuler explicitement de nombreux paramètres matériels. Pour rendre cette tâche plus souple et sûre, SPOC (voir Annexe A) est un choix imposé dans le projet. Il consiste à une abstraction haut niveau de la programmation GPGPU. En simplifiant la programmation GPGPU et en autorisant le développement d'optimisations supplémentaires. SPOC nous permet d'atteindre un haut niveau de performance. Il assure aussi la portabilité. Et fournit des squelettes qui peuvent être utilisées pour composer les Kernels². Les squelettes

2. Dans la programmation GPU, une fonction définie par l'utilisateur qui tourne sur le GPU est appelé un Kernel.

sont des constructions algorithmiques basées sur des patrons de conception communs, qui peuvent être paramétrées pour adapter leur comportement. Les squelettes décrivent explicitement les relations entre les kernels et les données. En utilisant cette information, il est possible d'optimiser automatiquement le calcul global [1]. Spoc est un outil issue du travail de l'équipe APR du laboratoire LIP6 à l'Université Pierre et Marie Curie.

Chapitre 3

Contexte et objectifs

Ce chapitre décrit le contexte du stage, ces objectifs, ainsi que son plan général.

3.1 Contexte

Les simulations à grande échelle que l'on rencontre en calculs numériques sont difficiles à gérer. Cela provient de leur nature exploratoire qui nécessite d'évaluer un grand nombre de combinaisons de paramètres, le tout en lançant de nombreuses tâches en parallèle. Les SWfMS ont démontré leurs capacités dans différents domaines scientifiques comme l'astronomie et le biologie [13], [9], orchestrant la parallélisation des activités de flux de travail [3]. La programmation GPGPU présente une solution très intéressante dans la parallélisation des milliers d'activités indépendantes, comme dans les opérations de MapReduce. Cependant, il y a plusieurs défis ouverts dans l'exploration des différents modèles de parallélisme pendant l'exécution du flux de travail. Parmi les défis à relever, on s'intéresse ici au problème de l'optimisation de la planification d'exécutions parallèles, laquelle doit tirer profit des architectures hétérogènes (multi-cœurs, accélérateurs). Par exemple la programmation des cartes graphiques demande un couplage fort entre les unités de calculs parallèles du GPU avec le CPU pour obtenir de bonnes performances, en particulier sur le transfert de données. Les systèmes hétérogènes nécessitent des conceptions complexes combinant différents paradigmes de programmation pour gérer chaque matériel de manière spécifique.

On cherche alors à abaisser cette complexité dans les flux de travail pour calculs numériques parallèles en utilisant des abstractions de flux de données et des constructions de haut niveau de programmation parallèle pour représenter la spécification du flux de travail et permettre d'optimiser le plan d'exécution parallèle.

3.2 Objectifs

Dans ce projet nous visons à aborder l'efficacité dans l'exécution parallèle d'un flux de travail de calcul numérique. En laissant plus d'espace au moteur de gestion de flux de travail pour prendre les décisions sur les choix des plans d'exécutions. Ce travail en commun contribue en combinant les abstractions de programmation de différents niveaux, à savoir au niveau de la langage de spécification de flux de travail (groupe brésilien) et au niveau de la programmation parallèle (groupe français). la sémantique des opérations algébriques fournit l'exécution en parallèle en mappant de la langue de flux de travail vers des expressions algébriques équivalentes, nous prévoyons que l'expression peut encore être mise en correspondance avec des abstractions de SPOC. De cette façon, le mapping vers SPOC profitera de la sémantique des opérateurs algébriques. Par exemple, en fonction de l'exécution d'un Map, un Réduire, un Filter ou d'autres ensembles des opérations la meilleure abstraction parallèle est choisie pour être mise en correspondance avec le matériel parallèle spécifique. L'objectif est donc de combiner la sémantique des opérations algébriques de dataflow, comme celles proposées dans le système Chiron, avec la puissance de construction de composition de squelettes SPOC pour obtenir la génération dynamique de l'ordonnancement des exécutions parallèles. Cette combinaison, en plus d'être complémentaire, est originale dans le cadre de flux de travail pour calculs numériques. Ensemble, ils ont le potentiel d'isoler le matériel et la programmation de bas niveau de la spécification de haut niveau du flux de travail. Les abstractions prévoient également l'application des règles d'optimisation génériques.

Il existe plusieurs encapsulations de Chiron (voir Annexe A) dans des différents domaines (Clusters, Cloud-based...). Nous visons à développer une nouvelle encapsulation de Chiron qui supporte l'accélération GPU pour l'exécution parallèle de workflow. Le traitement parallèle dans Chiron est obtenu dans un style MapReduce (Hadoop). La flexibilité amenée par ce concept algébrique nous permet d'utiliser un mode d'activation d'activité qui nous permet d'ordonnancer dynamiquement le flux de travail et de l'optimiser. Dans ce mode les activités des flux de travail sont présentées par des activations (des programmes informatiques externes) [12]. On cherche dans un première temps à déployer ces activations en parallèle sur le GPU, et à étudier l'algorithme de l'optimisation et de la planification d'exécution parallèle de ces activations, à cause de la grande taille des données analysées, et l'intensité du calcul, la deuxième étape sera d'effectuer cette exécution parallèle sur plusieurs GPUs. Une étape envisagée en fonction de l'avancement du stage sera d'étudier la possibilité de parallélisation des fonctionnalités de Chiron, l'objectif sera de combiner les fonctionnalités de Chiron et celles de SPOC dans une nouvelle extension de Chiron pour l'environnement HPC.

3.3 Planification du travail

Le stage est organisé selon le plan général suivante :

3.3.1 Planification du travail

- Février 2016 :
 1. État de l'art.
 2. Sélectionner et installer la plateforme de travail.
- Mars 2016 :
 1. État de l'art.
 2. Analyser et étudier Chiron.
 3. S'initier à la programmation fonctionnelle, impérative, et orientée objet avec Ocaml.
- Avril 2016 :
 1. Exécuter des flux de travaux de Chiron sur la carte graphique.
 2. Exécuter des flux de travaux de chiron sur multi-gpu.
- Mai 2016 :
 1. Réflexion sur l'algorithme de l'optimisation et de la planification d'exécution parallèle.
 2. Mise en œuvre d'une version fonctionnelle de Chiron.
- Juin 2016 :
 1. Phase d'expérimentations.
 2. Comparaison entre les deux approches (Chiron de base, Chiron amélioré).
- Juillet 2016 :
 1. Rédaction.

3.3.2 Prospective

Étudier la possibilité de parallélisation des opérations de Chiron, pour introduire une nouvelle version de Chiron qui consiste à combiner les fonctionnalités de Chiron et celles de SPOC.

Chapitre 4

Cas tests

Dans ce chapitre nous illustrons quelques exemples sur l'exécution parallèle des flux du travail scientifiques.

4.1

4.1.1 Opération de calcul

On commence par un exemple simple pour valider la combinaison entre Chiron et Spoc. Il s'agit d'une opération d'addition entre les contenus des fichiers d'un format csv représentant des données tabulaires.

cet exemple montre l'utilité de chiron d'automatiser des operation de calcul sur un grand nombre des fichiers .

Les différents fichiers présents

- **configuration des données**

Le fichier input.dataset qui organise le flux de données afin de le traiter par Chiron. C'est un fichier de format "csv". Il contient trois colonnes ID, FILE1, FILE2. L'identifiant ID indique un flux de travail à appliquer sur les champs suivants. Les champs FILE1 et FILE2 contiennent les chemins des fichiers des données à traiter.

- L'invocation d'un programme SPOC

Pour chaque activité existe un fichier de commande extractor.cmd qui fait le lien avec Chiron en invoquant le programme Spoc implémenté.

- la configuration de flux de travail

Le fichier XML Chiron.xml contient la description de flux du travail

Input relation "iact1" :

la relation iact1 est la relation d'entrée de l'activité act1. ID, FILE1, FILE2.

Output relation "oact1" :

la relation oact1 est la relation produite par l'activité act1. ID, FILE1, FILE2, FILE3. FILE3 : un champ ajouté par l'extracteur présente la sortie de l'opération appliquer sur FILE1 et FILE2.

- Add1_Chiron_vecteurs.byte

Le programme de Spoc prend en arguments ID FILE1 FILE2. Il applique une opération d'addition entre la colonne V de fichier FILE1 et la colonne K de fichier FILE2, et met le résultat dans un fichier de sortie. Et Il ajoute dans la relation de sortie une colonne FILE3 qui contiennent les chemins de fichiers de sorties.

Dans la base de données, pour "File_Workflow" on trouve deux tables Iact1 et Oact1.

La table iact1 contient les champs de la relation d'entrée.

La table oact1 contient les champs de la relation de sortie, on trouve la colonne FILE3 qui contient les chemins des fichiers de sorties.

4.1.2 Canny détection des contours

Dans cet exemple nous avons choisi d'appliquer le filtre de Canny. Nous avons opté ce choix car il illustre bien le flux de travail avec plusieurs activités qui s'enchaînent d'une façon que la sortie d'une activité est une entrée d'une autre.

Le filtre de Canny est utilisé en traitement d'images pour la détection des contours. L'entrée est une image en gris et la sortie est une image en noir et blanc avec une ligne blanche d'une largeur d'un pixel indiquant les bords. Un bord peut être défini comme un lieu de contraste élevé.

L'algorithme de Canny est conçu pour être optimal suivant trois critères clairement explicités :

Bonne détection : faible taux d'erreur dans la signalisation des contours. Bonne localisation : minimisation des distances entre les contours détectés et les contours réels.

Clarté de la réponse : une seule réponse par contour et pas de faux positifs.

Il comporte quatre étapes : la réduction du bruit, le gradient d'intensité, la direction des contours, la suppression des non-maxima et le Seuillage des contours. Implémenter respectivement par les filtres suivants flou gaussien, filtrage de sobel, suppression non maximale, le seuillage hystérésis.

On décrit ci-dessous brièvement chaque étape.

flou gaussien Dans la première étape un flou gaussien est effectué pour réduire le bruit de l'image originale avant d'en détecter les contours.

Cela est nécessaire parce que le bruit est généralement un contraste élevé et conduirait donc à des faux positifs. Il est implémenté en utilisant la convolution de l'image. La convolution de l'image est une opération qui remplace essentiellement chaque pixel avec une moyenne pondérée de ses voisins.

Sobel filtre et direction des contours Après le filtrage, l'étape suivante est d'appliquer un gradient qui retourne l'intensité des contours. le filtrage de Sobel remplace chaque pixel par une dérivée de x et y d'une combinaison des pixels voisins. Ce faisant, les pixels dans les zones de contraste élevé seront plus brillant que les pixels dans les zones de contraste faible. Ceci trouve essentiellement les zones où les bords sont plus susceptibles d'exister, mais il ne repérer pas précisément où les bords sont. Comme pour le flou gaussien, cela se fait en utilisant l'image convolution, mais la convolution est effectuée sur deux fois : une fois pour le dérivé de x et une fois pour le dérivé de y. Le pixel est alors remplacé par l'essentiel :

où di représente le changement d'intensité. Au cours de cette étape, la direction du gradient est également calculé pour chaque pixel qui est nécessaire pour une suppression non maximale. Les orientations des contours sont déterminées par la formule :

Nous obtenons finalement une carte des gradients d'intensité en chaque point de l'image accompagnée des directions des contours.

Suppression des non-maximum La carte des gradients obtenue précédemment fournit une intensité en chaque point de l'image. Une forte intensité indique une forte probabilité de présence d'un contour. Toutefois, cette intensité ne suffit pas à décider si un point correspond à un contour ou non. Seuls les points correspondant à des maxima locaux sont considérés comme correspondant à des contours, et sont conservés pour la prochaine étape de la détection. Un maximum local est présent sur les extrema du gradient, c'est-à-dire là où sa dérivée s'annule.

suppression non maximale exclut les pixels qui font partie d'un bord, mais ne définissent pas le bord. Le résultat est que nous condons ces bords de gradient de large en une seule ligne de 1 pixel. Notez que le résultat produit est toujours pas le produit fini, ces lignes ne sont pas purement blanc et ne signifient pas nécessairement un avantage, ils représentent encore la place une probabilité d'un bord. la suppression de non-maximale est effectuée en utilisant la direction du gradient trouvé à l'étape précédente et en comparant le pixel courant avec des pixels voisins de chaque côté. Si le pixel est plus faible en intensité que l'un de ces pixels voisins, alors il est pas considéré être le bord vrai, si sa valeur est remplacée par 0. Si le pixel est la plus forte intensité de ses voisins dans la direction du gradient, puis il peut être le bord vrai, si sa valeur est conservée.

Seuillage des contours La différenciation des contours sur la carte générée se fait par seuillage à hysteresis. Cela nécessite deux seuils, un haut et un bas ; qui seront comparés à l'intensité du gradient de chaque point. Le critère de décision est le suivant. Pour chaque point, si l'intensité de son gradient est inférieur au seuil bas, le point est rejeté, et si il est supérieur au seuil haut, le point est accepté comme formant un contour. Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté. Une fois ceci réalisé, l'image obtenue est binaire avec d'un côté les pixels appartenant aux contours et les autres.

ci desous l'implémentation de l'algorithme de Canny détection de bord en utilisant CHI-

RON et Spoc.

Le flux de travail On précise la structure de flux de travail dans le fichier XML. Le workflow est constituées de 5 taches, chaque tache correspond à un filtre.


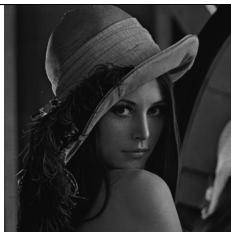




				
	filtre	input	output	
act1	gray	image	oact1	
act2	Gaussian	image1	image2	
act3	Sobel	2.3	image3	
act4	Non-max	image3	image4	
act5	hysteris	mage4	image5	

TABLE 4.1 – Le flux de travail de canny

Logiciels

Durant ce stage on s'appuie principalement sur deux logiciels :

Chiron <http://chironengine.sourceforge.net/>.

Chiron est un système de gestion de flux de travail de calculs numériques (*scientifique workflow management system*), il exécute ces simulations comme une chaîne d'activités (programmes) et un flux de données (dataflow) sur ces activités. Ce système fournit la gestion des simulations scientifiques, leur exécution parallèle tout en enregistrant la provenance des données. Chiron implémente l'approche algébrique dans un style MapReduce. L'utilisation de MapReduce comme approche de programmation permet aux scientifiques de programmer d'une façon plus simple la procédure du calcul en cachant le parallélisme, qui peut être complexe à gérer [12].

SPOC <http://www.algo-prog.info/spoc/>.

SPOC (Stream Processing with OCaml) consiste en une extension à OCaml associée à une bibliothèque d'exécution. L'extension permet la déclaration de noyaux GPGPU externes utilisables depuis un programme OCaml, tandis que la bibliothèque permet de manipuler ces noyaux ainsi que l'automatisation des transferts de données nécessaires à leur exécution. SPOC offre, de plus, une abstraction supplémentaire en unifiant les deux environnements de développement GPGPU (Cuda et OpenCL) en une même bibliothèque [2], [1].

Bibliographie

- [1] Mathias Bourgoïn and Emmanuel Chailloux. Gpgpu composition with ocaml. In *Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*, page 32. ACM, 2014.
- [2] Mathias Bourgoïn, Emmanuel Chailloux, and Jean-Luc Lamotte. Efficient abstractions for gpgpu programming. *International Journal of Parallel Programming*, 42(4) :583–600, 2014.
- [3] Marc Bux and Ulf Leser. Parallelization in scientific workflow management systems. *CoRR*, abs/1303.7195, 2013.
- [4] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails : visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.
- [5] Peter Couvares, Tevfik Kosar, Alain Roy, Jeff Weber, and Kent Wenger. Workflow management in condor. In *Workflows for e-Science*, pages 357–375. Springer, 2007.
- [6] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. Pegasus : A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3) :219–237, 2005.
- [7] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks : A survey. *Computing in Science & Engineering*, 10(3) :11–21, 2008.
- [8] Scott J. Krieder, Justin M. Wozniak, Timothy Armstrong, Michael Wilde, Daniel S. Katz, Benjamin Grimmer, Ian T. Foster, and Ioan Raicu. Design and evaluation of the gemtc framework for gpu-enabled many-task computing. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 153–164, New York, NY, USA, 2014. ACM.

- [9] Marta Mattoso, Jonas Dias, Kary ACS Ocaña, Eduardo Ogasawara, Flavio Costa, Felipe Horta, Vítor Silva, and Daniel de Oliveira. Dynamic steering of hpc scientific workflows : A survey. *Future Generation Computer Systems*, 46 :100–113, 2015.
- [10] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2) :40–53, March 2008.
- [11] Eduardo Ogasawara, Jonas Dias, Daniel Oliveira, Fábio Porto, Patrick Valduriez, and Marta Mattoso. An algebraic approach for data-centric scientific workflows. *Proc. of VLDB Endowment*, 4(12) :1328–1339, 2011.
- [12] Eduardo Ogasawara, Jonas Dias, Vítor Silva, Fernando Chirigati, Daniel Oliveira, Fábio Porto, Patrick Valduriez, and Marta Mattoso. Chiron : a parallel engine for algebraic scientific workflows. *Concurrency and Computation : Practice and Experience*, 25(16) :2327–2341, 2013.
- [13] Daniel Oliveira, Eduardo Ogasawara, Kary Ocaña, Fernanda Baião, and Marta Mattoso. An adaptive parallel execution strategy for cloud-based scientific workflows. *Concurrency and Computation : Practice and Experience*, 24(13) :1531–1550, 2012.
- [14] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3) :31–36, 2005.

Table des matières

1	Introduction	1
2	État de l’art	3
2.1	Les systèmes de gestion de flux de travail scientifique	3
2.2	La provenance des données	4
2.3	L’approche algébrique	4
2.4	Le parallélisme	5
2.5	La programmation GPGPU	6
3	Contexte et objectifs	9
3.1	Contexte	9
3.2	Objectifs	10
3.3	Planification du travail	11
3.3.1	Planification du travail	11
3.3.2	Prospective	11
4	Cas tests	13
4.1	13
4.1.1	Opération de calcul	13
4.1.2	Canny détection des contours	14
A	Logiciels	19