Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet ASR

2017-2018

Configuration d'un OS pour le calcul parallèle

Tuteur académique
Patrick Martineau

Étudiants
Benjamin Caldas (DI5)
Logan Verecque (DI5)

13 janvier 2018

Liste des intervenants

Nom	Email	Qualité
Benjamin Caldas	benjamin.caldas@etu.univ-tours.fr	Étudiant DI5
Logan Verecque	logan.verecque@etu.univ-tours.fr	Étudiant DI5
Patrick Martineau	patrick.martineau@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Benjamin Caldas et Logan Verecque susnommés les auteurs.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Patrick Martineau susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

Les auteurs reconnaissent assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

Les auteurs attestent que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

Les auteurs attestent ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

Les auteurs attestent que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

Les auteurs reconnaissent qu'ils ne peuvent diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

Les auteurs autorisent l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Benjamin Caldas et Logan Verecque, *Configuration d'un OS pour le calcul parallèle*, Projet ASR, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
    author={Caldas, Benjamin and Verecque, Logan},
    title={Configuration d'un OS pour le calcul parallèle},
    type={Projet ASR},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2017-2018}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V2.2.5 (2017/05/02)

Table des matières

Table des figures

Table des figures

Première partie

Introduction

1 Contexte

Dans le cadre de notre cursus nous avons l'occasion de nous spécialiser en choisissant la voie des Systèmes d'Informations ou la voie d'Architecture, Système et Réseaux. Ce document présente le projet d'Architecture, Système et Réseaux réalisé par le binôme Benjamin Caldas / Logan Verecque finalisant la spécialisation ASR au sein de la 5ème année du diplôme d'Ingénieur en Informatique de l'école Polytechnique de Tours.

2 Sujet

La figure suivante contient l'énoncé exact de notre sujet.

Sujet ASR #1 : Mise en place d'une infrastructure de calcul parallèle - P. Martineau

On propose d'intégrer les différents outils de calcul parallèle dans un OS minimal qui pourrait être porté sur une clef USB. Cuda permet de profiter du parallélisme sur les GPU OpenMP permet de profiter du parallélisme entre les core d'un CPU MPI permet de profiter du parallélisme entre ordinateurs au sein d'un réseau.

Les trois outils sont complémentaires. On souhaite donc :
-prendre en main chaque outil
-intégrer leur utilisation et leur complémentarité (intégration) au sein d'un unique OS.
-prévoir de faciliter le déploiement de cet OS sur un ensemble de machines équipées ou non du matériel nécessaire pour mettre en œuvre chaque niveau de parallélisme
-> le but est de rendre indépendant le développement de l'exécution des processus. Cette exécution bénéficiant ou non de la parallélisation.

Figure 1 – *Énoncé exact du sujet*

3 Objectifs

L'objectif de notre projet était de mettre en place un système d'exploitation Unix contenant tous les outils permettant de réaliser du développement suivant les trois grands axes de parallélisations :

- OpenMP
- Nvidia GPU Cuda
- MPI

Ce système devait également être portable et bootable depuis n'importe quel clé usb. Cette contrainte indique également que le système doit être léger, simple d'utilisation et d'installation.

Deuxième partie

Définition du sujet

4 Système d'exploitation

Un système d'exploitation (SE), ou Operating System (OS) en anglais est un ensemble de programmes qui pilote tous les composants de l'ordinateur. Il dirige l'utilisation des ressources par les applications. Les ressources de stockage, de calcul et de communication sont donc gérer par le SE et distribuées aux différents logiciels. Le SE assure donc le lien entre le matériel informatique et les les applications utilisées ou non par les utilisateurs. Ces différentes interactions sont illustrées dans la figure suivante :

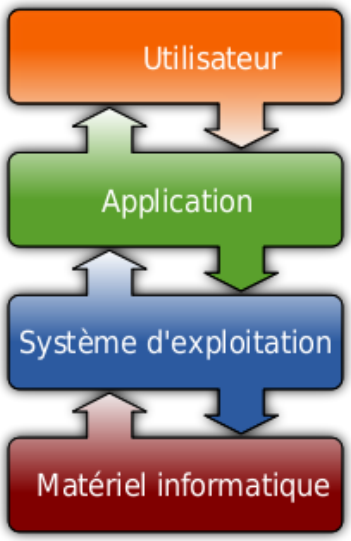

Figure 2 – *Le système d'exploitation dans l'architecture d'un ordinateur*

Les principaux systèmes d'exploitations sont Windows, Mac OS et les distributions de type Linux. Nous opterons pour une distribution de ce dernier car leur principal avantage est que tout est paramétrable facilement et rapidement. Ceci nous sera très utile dans le cadre de notre projet d'architecture systèmes et réseaux.

5 Clé USB bootable

Une clé USB bootable est une clé USB qui intègre un système capable de faire démarrer un système d'exploitation sans faire appel à un autre élément de stockage. Cela permet notamment d'installer un SE sur un ordinateur sans avoir le CD d'installation ou de pouvoir utiliser une distribution particulière depuis n'importe quelle machine.

Le GRUB ou (GRand Unified Bootloader) est un logiciel permettant de charger un système d'exploitation. Il permet d'amorcer les systèmes de la norme POSIX dont Linux. Ce logiciel joue donc un rôle important dans la création d'une clé USB bootable embarquant un SE de type Linux.

6 OpenMP

OpenMP (Open Multi-Processing) est une interface de programmation pour le calcul parallèle sur architecture à mémoire partagée. Cette API est multiplateforme (GNU/Linux, OS X et Windows) pour les langages de programmation C, C++ et Fortran. OpenMP se présente sous la forme d'un ensemble de directives, d'une bibliothèque logicielle et de variables d'environnement. De plus, OpenMP est portable et dimensionnable.

OpenMP permet donc de développer rapidement des applications parallèles à petite granularité en restant proche du code séquentiel.

Par ailleurs, il est également possible de réaliser de la programmation parallèle hybride en utilisant à la fois OpenMP et MPI.

7 Cuda

CUDA (Compute Unified Device Architecture) est une technologie en GPGPU, c'est-à-dire utilisant un processeur graphique (GPU) pour exécuter des calculs à la place du processeur (CPU). L'idée est donc de permettre de « soulager » le processeur en utilisant le processeur graphique pour effectuer un maximum de calcul en parallèle de celui-ci.

CUDA permet de programmer des GPU en C. Elle a été développée par NVidia pour ses cartes graphiques et garde donc l'exclusivité autour de cette marque.

8 MPI

MPI (Message Passing Interface) est une bibliothèque de fonctions, conçue en 1993, permettant le calcul parallèle en C, C++ et en Fortran. Elle exploite les ordinateurs multiprocesseur, donc quasiment toutes les machines actuelles, qu'elles soient en réseau ou non. Elle s'appuie sur cette communication en réseau pour distribuer les calculs à effectuer entre toutes les machines du réseau par passage de messages.

Lorsqu'une application sera exécuté avec MPI, elle sera clonée suivant le nombre de processus spécifié et envoyée sur les différentes ressources de stockage et ceci par l'intermédiaire du réseau, comme le montre la figure suivante :

MPI utilise le concept de communicateurs. Un communicateur est un ensemble de processus pouvant communiquer ensemble et cette communication sera possible seulement si ces processus sont dans le même communicateur.

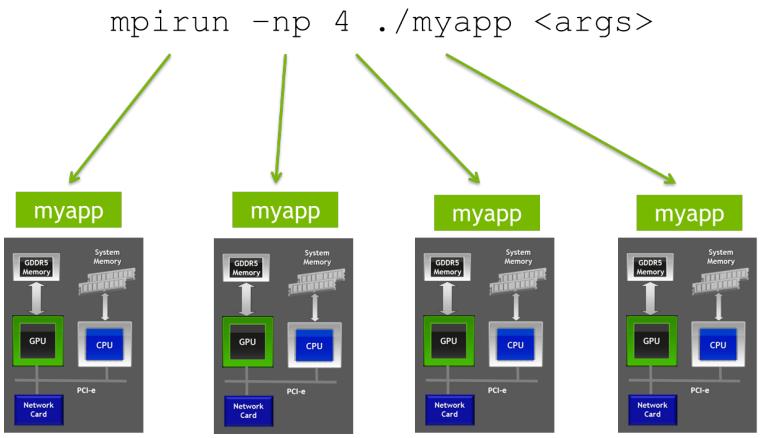

Figure 3 – *Application utilisant MPI*

Troisième partie

Solution envisagée

9 Principe

L'idée est de s'appuyer sur un OS le plus léger possible pour être portable et bootable directement depuis une clé USB. Néanmoins, il nous faut un système capable de supporter les trois outils de parallélisations. Seulement, si OpenMP et MPI sont disponible et utilisable sur l'intégralité du monde Unix ce n'est pas le cas de Cuda qui se limite seulement à quelques système d'exploitation (cf. 10. Choix du système d'exploitation).

Une fois notre clé disponible et le système correctement installé il nous faut mettre en place et installer sur celle-ci nos trois outils de développement. Nous commencerons tout d'abord par OpenMP puis par CUDA et enfin par MPI. A noter que chaque phase d'installation fera intervenir une phase de tests afin de valider le bon fonctionnement de notre outil avant de passer à l'installation de la suivante.

Enfin, l'utilisation des trois outils est effective dans la limite des conditions suivantes : - OpenMP : Toujours - CUDA : Si une carte graphique NIVIDIA est présente et disponible - MPI : Multi-machines si réseau disponible

10 Choix du système d'exploitation

Cette section présente la démarche de sélection du système d'exploitation.

10.1 Recherche

Afin de choisir notre système d'exploitation nous avons cherché à trouver un système d'exploitation très léger capable de respecter les contraintes fixées pour ce projet. Parmi ces contraintes, la plus importante était évidemment d'être capable de supporter CUDA. Pour cela, NVIDIA a mis à disposition sur son site officiel la liste des systèmes capables de l'utiliser. Cette liste est la suivante :

Nous avons ensuite regardé de plus près chacun des systèmes et nous avons pu observer que le plus léger était Ubuntu. Néanmoins, il était encore possible d'aller encore plus loin et de s'intéresser à la famille de système d'exploitation Ubuntu elle-même.

Table 1. Native Linux Distribution Support in CUDA 9.1

Distribution	Kernel	GCC	GLIBC	ICC	PGI	XLC	CLANG
x86_64							
RHEL 7.x	3.10	4.8.5	2.17	17.0	17.x	NO	4.0.0
RHEL 6.x	2.6.32	4.4.7	2.12				
CentOS 7.x	3.10	4.8.5	2.17				
CentOS 6.x	2.6.32	4.4.7	2.12				
Fedora 25	4.8.8	6.2.1	2.24-3				
OpenSUSE Leap 42.3	4.4.68	4.8.5	2.22				
SLES 12 SP3	4.4.21	4.8.5	2.22				
Ubuntu 17.04	4.9.0	6.3.0	2.24-3				
Ubuntu 16.04	4.4	5.3.1	2.23				
POWER8(*)							
RHEL 7.x	3.10	4.8.5	2.17	NO	17.x	13.1.6	4.0.0
Ubuntu 16.04	4.4	5.3.1	2.23	NO	17.x	13.1.6	4.0.0
POWER9(**)							
Ubuntu 16.04	4.4	5.3.1	2.23	NO	17.x	13.1.6	4.0.0
RHEL 7.4 IBM Power LE	4.11	4.8.5	2.17	NO	17.x	13.1.6	4.0.0

(*) Only the Tesla GP100 GPU is supported for CUDA 9.1 on POWER8.

(**) Only the Tesla GV100 GPU is supported for CUDA 9.1 on POWER9.

Figure 4 – *Liste des SE compatible avec Cuda*

10.2 Lubuntu

Notre choix s'est donc finalement porté sur Lubuntu pour plusieurs raisons importantes. Lubuntu est un projet de distribution GNU/Linux et une version dérivée d'Ubuntu. L'objectif de Lubuntu est d'être une version plus légère, plus économe en ressources matérielles et moins consommatrice en énergie qu'Ubuntu.

On notera que Lubuntu utilise pour cela l'environnement de bureau LXDE et le compilateur GCC n'est pas installé par défaut.

Quatrième partie

Étapes de réalisation

Dans cette partie, nous verrons quelles étaient les différentes étapes de réalisation du projet.

11 Création de la clé bootable

Il a fallut, dans un premier temps, créer la clé bootable. Ceci s'est fait en deux étapes, premièrement la création de clé bootable depuis laquelle on allait lancer l'installation et ensuite le partitionnement de la clé USB et l'installation du SE.

11.1 Création de l'environnement

Pour partitionner la clé USB et y installer un SE, il faut s'appuyer sur une distribution Linux. C'est pour ceci qu'il a fallut créer une clé Live Lubuntu. On a utilisé une clé USB de faible capacité ici, 4 Go suffisent. Nous avons utilisé l'outil Linux live USB creator en spécifiant un *.iso* Lubuntu. La création de la clé USB bootable en mode live a prit environ 15 minutes. Nous avons ensuite démarré notre machine sur cette clé USB live.

11.2 Partitionnement de la clé USB et Installation du SE

Après avoir booté sur la clé USB live tout juste créé, nous avons lancer l'utilitaire d'installation du SE Lubuntu. Nous avons précisé que nous souhaitions une installation personnalisée. Pour installer Lubuntu en version persistante dans le temps, il faut prévoir une clé de 16 Go mininum. Nous avons ensuite choisis le lecteur sur lequel effectué l'installation : notre clé USB puis nous avons spécifié de donner 4 Go pour le swap et le reste (environ 10 Go) pour les données en ext2. Nous avons choisis cette même clé USB pour l'installation du programme de démarrage, aussi appelé GRUB. L'installation a ensuite durée environ 2 heures. Après ceci, notre clé USB était fin prête.

12 Gcc

12.1 Installation

Le paquet gcc n'est pas présent nativement lorsque l'on installe Lubuntu sans les outils recommandés, il a donc fallut l'installer. Dans ce but, nous avons effectuer une mise à jour des services *apt-get* puis nous avons installer le paquet gcc à l'aide de la commande *apt-get install gcc*. L'installation de ce paquet d'environ 80 Mo prend un peu plus d'1 heure.

12.2 Validation OpenMP

Afin de tester si gcc a bien été installé, on a vérifié en entrant la commande permettant de connaître sa version. La commande est *gcc –version*.

Lorsque gcc est installé, OpenMP est utilisable. On a donc codé un programme simple permettant de voir si la programmation multi-coeurs est bien effective.

13 Cuda

13.1 Vérifications

Avant d'installer les paquets nécessaires au fonctionnement de Cuda, il y a plusieurs vérifications à effectuer dans le but de savoir si le système est correctement paramétré pour exécuter des programmes mettant en oeuvre la programmation parallèle sur GPU.

Il a fallut dans un premier temps vérifier que la distribution Linux était compatible. Vu que nous avons choisit le SE en fonction de ce critère de compatibilité, il n'y avait pas de soucis de ce côté. Ensuite, il fallait vérifier que l'ordinateur possède une carte graphique de marque NVidia. En revanche, dans notre cas, nous paramétrons une distribution embarquée sur clé USB, cette vérification sera donc a effectuer sur les postes qui utiliserons notre clé USB bootable. Enfin, Cuda s'appuie sur le paquet gcc que nous avons installé puis vérifié précédemment.

13.2 Installation Cuda

Nous avons ensuite pu installer les paquets nécessaire à Cuda en suivant les recommandations du site *developer.nvidia.com*. Cet installation requière environ 3 Go d'espace disque est prend approximativement 3h.

13.3 Installation nvcc

13.4 Validation

14 MPI

14.1 Installation

14.2 Validation mono-poste

14.3 Validation sur réseau

Cinquième partie

Gestion de projet

15 Organisation

Les séances dédiés à la réalisation des projets ASR étaient répartis à hauteur de 2 par semaine. Il était très difficile de se diviser les tâches car notre projet consistait en une suite d'installations suivies de validations et de sauvegardes. Il était impossible d'assurer deux installations en même temps car celles-ci devait s'effectuer l'une à la suite des autres. Pendant qu'une personne effectuait les installations, la seconde personne était souvent sur un travail de recherches d'informations afin de préparer au mieux la prochaine installation.

16 Planning

Lors de l'analyse du projet, nous avons identifié les grandes tâches à effectuer. Celles-ci sont illustrées dans la figure suivante :

Figure 5 – *Tâches identifiées au début du projet*

La réalisation de projet s'est déroulée sur 4 mois. Les différentes tâches et leurs répartitions temporelles sont illustrées dans le diagramme de Gantt suivant :

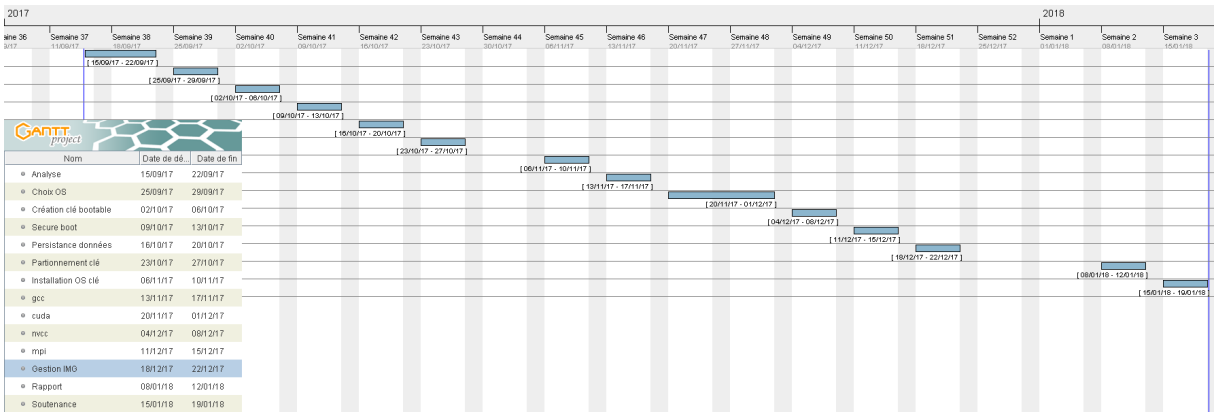

Figure 6 – *Planning du projet*

17 Gestion de versions

Dans le but de gérer les différentes versions de l'avancement de nos installations sur la clé USB, nous avons généré plusieurs fichiers *.img* dans le but de ne pas perdre notre avancement et de revenir en version antérieure en cas e problème majeur. Les différentes versions sont illustrées dans la figure suivante :

Figure 7 – *Versions des images*

Sixième partie

Problèmes et difficultés rencontrés

18 Connexion réseau de Polytech

L'une des deux difficultés majeures rencontrés lors de ce projet concerne la connexion wifi délicate de Polytech. Selon l'heure de la journée, sur nos machines personnelles respectives il était possible que la vitesse de téléchargement descende jusqu'à 30 Ko/s. Cette situation n'était donc pas simple à gérer lorsque nous avions par exemple un outil de plus d'un Go à récupérer. En conséquence, plus nous avons avancé vers la fin du projet et plus les différents téléchargements ont été effectués soit en parallèle des créneaux de projet ASR mais chez nous soit pendant notre temps libre.

19 Temps des installations

La deuxième difficulté qui nous aura contraint à revoir notre façon de travailler est le temps d'installation des différents outils. En effet, si on considère le débit Internet difficile de Polytech en wifi sur nos machines personnels le temps d'installation des trois outils a pu s'élever jusqu'à 6h au total. C'est l'installation de CUDA qui est la plus difficile au vu de la taille assez conséquente de l'outil. Evidemment, ce genre de difficulté a posé de sérieux soucis pendant les créneaux de seulement 2h dans l'emploi du temps. Il a donc fallu gérer les installations en dehors de Polytech et des différents créneaux quand la situation l'exigeait.

20 Secure Boot

Un autre problème est également survenu assez rapidement dans notre projet. Cela concerne la façon de booter sur la clé USB au démarrage. En effet, selon la marque et le modèle de l'ordinateur il nous était difficile de booter sur notre clé. La faute à cause de l'outil de « Secure Boot » par exemple sur les modèles d'ASUS qui bloque par défaut les démarrages de système d'exploitations par des périphériques extérieurs à l'ordinateur.

De plus, il faut veillez à démarrer sur la clé USB en utilisant le mode *Legacy* plutôt que le mode *UEFI* (Unified Extensible Firmware Interface ou interface micrologicelle extensible unifiée en français). Ce dernier va bloquer un certain nombre de fonctionnalités et le boot risque de pas aboutir alors que le mode Legacy n'a pas le même paramétrage et permettra de booter sur la clé USB sans soucis.

Il est important de noter que ce problème ne concerne pas les PC présents au sein de l'école. Le démarrage par clé USB se fait sans difficulté à partir du menu de sélection de démarrage.

21 Mode Live et mode persistant

Il existe trois façons de booter sur une clé USB :

- En mode live : on utilise le système d'exploitation mais aucune donnée ne sera sauvegardée. Le système sera remis à zéro au prochain démarrage de celui-ci. Ce mode est présent pour permettre l'installation complète du système sur la clé ou à un autre endroit.

- En mode persistant : on utilise le système d'exploitation avec une capacité de stockage alloué par défaut à 4Go maximum.

- OS installé : l'OS est définitivement installé sur la clé USB avec une capacité définie par l'utilisateur lors de l'utilisation.

Une idée au départ était d'utiliser le « Mode Persistant » pour installer nos outils sans avoir un installé de manière complète le système d'exploitation sur notre clé. Malheureusement, cela n'est pas possible car la taille de nos outils au total dépasse largement 4Go. Nous avons donc dû prendre en compte cela et nous avons décidé d'installer de manière complète le système sur la clé.

22 Limite de stockage

Comme dit précédemment, la taille de nos outils au total est assez élevée, presque 16Go au final. En effet, presque l'intégralité des 16Go disponible par la clé USB permet l'installation des différents outils, ce qui laisse très peu de place pour le développement en lui-même.

23 Fragilité

Il faut absolument manier la clé USB avec le plus grand soin. Comme tout périphérique extérieur contenant un système d'exploitation celui-ci est fragile. Il est important de ne pas être brutal avec celui-ci ni de retirer la clé en cours d'exécution du système ou si celui-ci n'est pas encore totalement terminé. Dans le cas contraire, on peut s'exposer à une destruction ou une corruption pur et simple du système d'exploitation. Dans cet optique, il est important d'utiliser des sauvegardes de notre OS pour éviter une réinstallation complète de celui-ci.

Septième partie

Outils utilisés

Dans cette partie, nous récapitulerons tous les outils utilisés, nous détaillerons leurs fonctionnalités et nous indiquerons où et comment se les procurer.

24 Linux Live Creator

LinuxLive USB Creator est un logiciel gratuit et open source pour Windows qui permet de créer une clé USB bootable avec une distribution Linux dessus. Ce logiciel se veut très simple d'utilisation avec une interface ergonomique et intuitive. Chaque étape nécessaire à la création de la clé USB est affublé d'un feu de circulation permettant d'indiquer à l'utilisateur si l'étape a bien été remplie.

Cet outil est disponible sur le site officiel du logiciel : http ://www.linuxliveusb.com/fr/

25 USB Image Tools

Sauvegarde des .img

26 Etcher

Ré-injection des .img

27 Diskpart

Diskpart est un outil logiciel de partionnement de disques durs informatique. Il est nativement intégrer sous Windows. Diskpart permet de découper un disque dur ou autre périphérique externe en plusieurs partitions, pour permettre par exemple la cohabitation de plusieurs systèmes d'exploitation sur une même machine. Diskpart permet également de préparer une clé USB en gérant les partitions systèmes, c'est ce qui nous intéresse ici.

Figure 8 – *Linux live USB creator*

Figure 9 – *Diskpart - Exécution et liste des commandes*

Huitième partie

Guide d'utilisation

Dans le but d'aider les utilisateurs à utiliser notre clé USB bootable intégrant un SE capable d'effectuer de la parallélisation avec OpenMP, Cuda et MPI, nous avons réalisé un guide d'utilisation.

Brancher la clé USB.

Ouverture de session "password" pour le compte polytech

Dire où sont les exemples

Commande OpenMP

Script check carte graphique pour Cuda

Commande Cuda

Commande MPI

Neuvième partie

Guide d'installation

Dans le but d'aider un utilisateur à installer l'image de notre SE capable d'effectuer de la parallélisation avec OpenMP, Cuda et MPI, nousa vons réalisé un guide d'installation. Ce guide eprmettra aussi à l'utilisateur de générer sa propre image après avoir fait ce qu'il voulait sur le SE. On lui indiquera également comment repartitionner sa clé USB.

Injection .IMG avec Etcher

Sauvegarde .IMG avec USB Image Tools

Rapartionnement avec Diskpart

Dixième partie

Guide du développeur

Dans le but d'aider un futur développeur qui se lancerait dans la même tâche que celle qui nous a été confiée, nous avons créé un guide destiné à une personne ayant des compétences assez avancées en système dans le but de lui donner la marche à suivre dans la réalisation du clé USB bootable intégrant un SE capable d'effectuer de la parallélisation avec OpenMP, Cuda et MPI.

Onzième partie

Conclusion

Livrables :
OS sur clé avec 3 méthodes de parallélisation
fichiers ajoutés dedans seront persistants

Configuration d'un OS pour le calcul parallèle

Résumé

Projet ASR

Mots-clés

Système d'exploitation, OpenMP, Cuda, MPI

Abstract

ASR project

Keywords

Operating system, OpenMP, Cuda, MPI

Tuteur académique

Patrick Martineau

Étudiants

Benjamin Caldas (DI5)

Logan Verecque (DI5)