UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT630 – Processus concurrents et parallélisme

Projet Partie 1

Travail présenté

à

M. Daniel-Junior DUBÉ

Par

William Gingras – 16104154

Olivier Perreault - 16212377

Le 28-06-2019

Table des matières

Introduction 1

Description du problème de la forge 1

Stratégie de résolution du problème de la forge 2

Aspects techniques 3

Approximation de notre modèle. 4

Hypothèse 4

Problème potentiel 4

Conclusion 4

Références 5

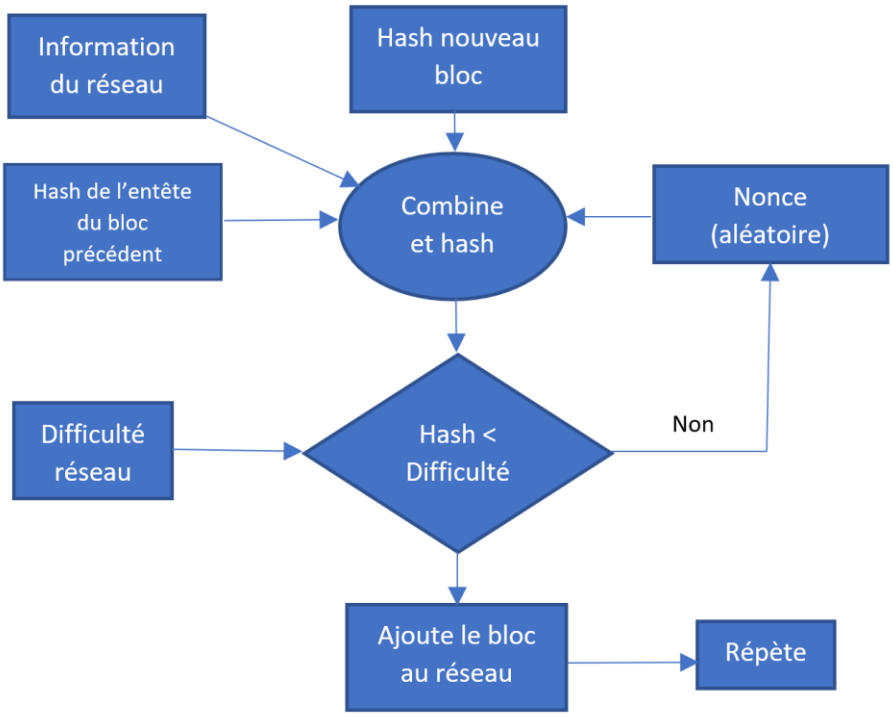
Introduction

Le réseau bitcoin a fait plusieurs fois la manchette depuis sa création en 2008. Tout récemment, Bitfarms, une compagnie dans le domaine du minage de Bitcoin annonçait qu'elle allait de l'avant avec un nouveau centre de calcul à Sherbrooke avec un investissement prévu de 250M$ [1]. Un investissement d'une telle somme se justifie par l'acquisition de matérielle à application spécifique afin de résoudre le problème de la forge de nouveau bloc [2]. Notre projet consiste à comprendre et analyser la performance de différentes techniques afin de résoudre le problème du minage de bitcoin. Pour y parvenir, nous allons faire une simulation du réseau bitcoin et appliquer différentes techniques de calcul afin de forger de nouveaux blocs.

Description du problème de la forge

Tout d'abord, le rôle d'un mineur est de forger un bloc le plus rapidement possible et l'ajouter dans la chaîne. Le processus de minage s'effectue selon le diagramme suivant:

Figure 1: Procédé de création d'un nouveau bloc [4]



Un bloc est forgé lorsqu'un des mineurs calcul un entête de bloc valide. L'entête est un SHA256 qui est obtenue en suivant un ensemble de règles. Pour que le hash d'un bloc soit valide, il doit respecter un certain nombre de règles. Parmi celles-ci il y a que le hash résultant débute par

un certain nombre de 0, le premier octet doit être plus petit que la première valeur significative de l'indicateur de difficulté et que le hash obtenu respecte l'équation suivante : $Hash_{nouveau} = SHA256(Version + Hash_{ancien} + Hash_{Transactions} + Timestamp + Difficulté + Nonce)$

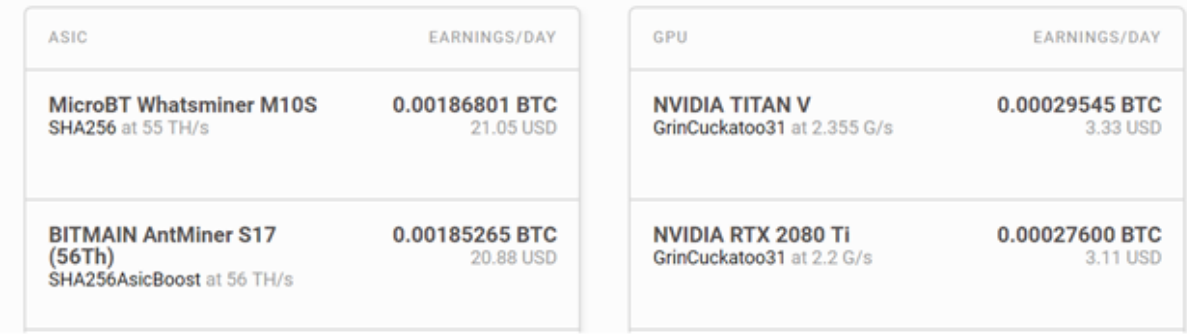
La version consiste à la version du réseau sur 4 Octets. Hash_ancien est le le SHA256 de l'entête du bloc précédent. Hash_transaction est le Hash des transactions du bloc courant obtenues à l'aide d'arbres de Merkel. Le Timestamp est en seconde depuis 1970. La difficulté correspond au nombre de 0 au début du bloc. Le Nonce est utilisé afin de respecter le critère précédent. Ainsi, un mineur essaie le plus nonce possible afin de trouver un hash qui respecte les règles précédentes. Le mineur qui réussit à miner le premier bloc reçoit une somme de bitcoin déterminer selon la version du réseau [3]. En date d'aujourd'hui cette somme est de 12.5 BTC.

Stratégie de résolution du problème de la forge

De nos jours, les mineurs ont tendance à combiner leurs efforts en formant des « pools ». En combinant leurs efforts, ils augmentent leurs chances de trouver un nonce valide et ils se séparent la récompense selon les règles du « pools ».

Pendant longtemps, le matériel le plus populaire pour miner des bitcoins était des GPU. Les GPU possèdent plusieurs « cores » d'où leurs popularités dans le calcul en parallèle. Cependant, ils sont de moins en moins populaire, car il existe un autre type de matériel plus efficace. Il s'agit des ASIC (« Aplication-specific integrated circuit »). Il s'agit de circuit intégré designer uniquement dans le but de calculer des fonctions SHA256. Comme le montre la Figure 1, les ASIC sont nettement supérieurs aux GPUs (55-56 TH/s vs 2.2-2.4 GH/s)

Figure 2: Comparaison ASIC – GPU [5]



Aspects techniques

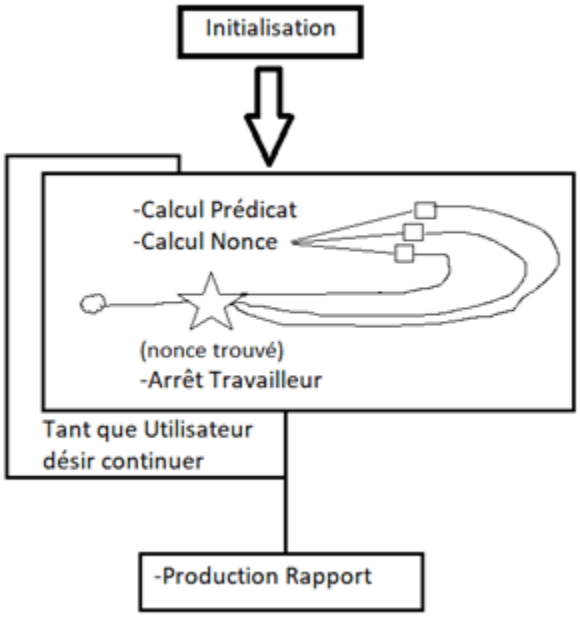
La parallélisation se fera au niveau des essaies des nonces afin de créer un bloc valide. Le mineur offrira les possibilités de miner en mode séquentiel, parallèle et GPU. La bibliothèque qui sera utilisée pour l'implémentation GPU sera OpenCL, car elle est compatible avec des cartes NVDIA et AMD contrairement à CUDA. Nous prévoyons programmer le mineur en C#, car il s'agit d'un langage de programmation complet et bien outillé pour résoudre un tel problème. L'analyse se penchera sur le temps de calculs des différents modes d'action selon différents niveaux de difficulté.

Le client va simuler une communication avec le réseau Bitcoin. De ce fait, il pourrait arriver qu'une autre personne sur le réseau mine le bloc. Ainsi, le mineur devra arrêter les travailleurs et passer au nouveau bloc. Cette fonctionnalité devra être optionnelle afin d'assurer l'homogénéité de l'analyse.

Malheureusement il nous sera impossible de comparer nos résultats afin du matériel de type ASIC, car nous ne disposons pas de tel équipement.

Le cycle de vie du mineur comprendra 3 phases visible sur le schéma suivant :

Figure 3: Cycle de vie du mineur



La phase d'initialisation sert à obtenir des informations sur l'environnement, effectuer des validations et initialiser les travailleurs. La seconde phase est la phase de travail qui consiste initialement à calculer le prédicat (un string qui contient toutes les informations du bloc excepté le nonce). Ensuite, le client répartit le calcul aux différents travailleurs et attend que soit un des

travailleurs trouve un nonce valide ou bien une autre personne sur le réseau en trouve un. Ensuite, il signale les travailleurs d'arrêter, calcul le nouveau prédicat et envoi une nouvelle charge aux travailleurs. Il répète ces étapes jusqu'à ce que l'utilisateur lui signale d'arrêter. Le client passera alors à la troisième phase, la production du rapport. Le rapport contiendra diverses informations utiles à l'analyse dont le temps moyen et le nombre de Hash/s (obtenue des travailleurs).

Approximation de notre modèle.

Afin de focaliser notre travail sur la parallélisation du mineur, nous allons apporter certaines simplifications au problème de la forge. Tout d'abord, la version du réseau sera statique. Le hash des transactions courantes sera généré aléatoirement. Aussi, le niveau de difficulté sera choisi par l'usager au lancement du mineur. Dans le réseau bitcoin, lorsque le mineur trouve un nonce valide il doit l'envoyer à un nœud du réseau, cependant cette étape est inutile à notre analyse.

Hypothèse

Nous croyons que le mineur sera le plus efficace sur GPU, car ce dernier possède le plus d'unités de calculs. Il est difficile d'estimer à l'avance le niveau de difficulté auquel le mineur va rencontrer un mur (le problème de la forge trop difficile).

Problème potentiel

Un des problèmes potentiels se trouve au niveau de l'implémentation logicielle de la simulation du réseau Bitcoin. Les simplifications du problème ont pour but de diminuer cette complexité. Ces simplifications n'auront pas d'impact sur la fiabilité des résultats. En théorie, si nous voulions obtenir des résultats qui représentent exactement le réseau Bitcoin il faudrait déployer notre mineur à même le réseau. Cependant, la difficulté du réseau est élevée à un point tel qu'il nous serait impossible de miner un bloc avec notre mineur.

Conclusion

En somme, le projet consiste à simuler un réseau bitcoin dans le but de comparer la performance de différents composants pour résoudre le problème de la forge. Les aspects de parallélisme se trouvent dans l'implémentation parallèle et GPU de la solution du problème et dans l'analyse des différences des performances. Si les performances sont exceptionnelles, va-t-on se lancer dans le minage de bitcoin?

Références

[1]
Dumas, J. (2019, March 18). Bitfarms amorce la construction de son centre de calcul à Sherbrooke. Retrieved from https://www.journaldemontreal.com/2019/03/18/bitfarms-amorce-la-construction-de-son-centre-de-calcul-a-sherbrooke

[2]
ASIC. (2019, April 24). Retrieved from https://en.bitcoin.it/wiki/ASIC

[3]
Mining. (2018, June 25). Retrieved from https://en.bitcoin.it/wiki/Mining#Reward

[4]

Mueller. (2018). HackerNoon – Why Bitcoin Fears Quantum Computers. Retrieved from https://hackernoon.com/why-bitcoin-fears-quantum-computers-and-iota-doesnt-697da531a11b

[5]

N. (2014). Profitability-calculator. Retrieved from https://www.nicehash.com/profitability-calculator