Cahier des charges : My Little Cluster

Cluster team

Table des matières

1 Présentation du groupe

1.1 Quentin de Laroussilhe

Je suis très motivé dans la réalisation ce projet, en effet, cela fait plusieurs mois que je m'intéresse aux techniques de parallélisme et je vais enfin me lancer dans une application concrête pour commencer à appréhender toutes les contraintes qu'imposent les architectures parallélisées.

FIGURE 1 – Quentin de Laroussilhe

1.2 Camille Dollé

1.3 Guillaume Sanchez

Cet energunene n'a qu'une seule idee en tete, biffler ses congeneres.

1.4 François Boiteux

Étudiant à EPITA en 2ème année de cycle préparatoire je décide cette année pour le projet libre du second semestre de rejoindre un groupe de projet qui s'interessera à la paralélisation ! C'est un domaine qui offre de multiples possibilités !

D'un côté personnel, j'ai vingt ans car j'ai effectué un an en faculté avant de rejoindre EPITA ! J'aime l'informatique, ma copine, mes amis et c'est ce qui compte le plus pour moi. Mes hobbies sont les jeux videos, la musique et les soirées.

Je m'occuperais de la partie serveur du projet ce qui me permettera d'en apprendre certainement plus sur les sockets et les fonctionnements server/client.

FIGURE 2 – boiteu_f

2 La parallélisation

2.1 Pourquoi paralléliser un calcul ?

Certains algorithmes peuvent demander des temps de calcul importants ou très important. Paralléliser un algorithme permet de distribuer les tâches entre plusieurs noeuds de calcul et ainsi diviser le temps nécessaire à sa réalisation.

Par exemple, un algorithme permettant de retrouver un mot de passe à base de sa signature nécessite de tester une à une toutes les combinaisons possibles jusqu'à trouver le hash recherché. Si la fonction de hashage est sûre, il n'existe pas de méthode de recherche plus optimisée. L'intérêt de diviser le calcul si l'on dispose de plusieurs machines pour effectuer un tel calcul est alors évident.

2.2 Les contraintes de la parrallélisation

Pour faire fonctionner un programme sur plusieurs clusters de calcul travaillant en collaboration on doit avoir pensé le programme pour un tel mode de fonctionnement.

2.2.1 Les effets de bords

Les algorithmes itératifs non designés pour fonctionner sur des architectures parallélisées contiennent des effets de bord, c'est à dire que l'opération d'une instruction executée à un instant t dépend des instructions éxecutées précédemment.

Ainsi, si nous possédons plusieurs noeuds de calcul et que nous décidions d'envoyer à chaque noeud une instruction différente aucun parallélisme ne serait possible car chaque noeud devrait attendre que les noeuds en charge des instructions précédentes aient fini d'executer leurs instructions. Cela revient strictement à une éxecution séquentielle des instructions.

La première difficulté est donc de minimiser les effets de bords pour minimiser les temps d'attente pour executer les instructions.

2.2.2 Les problématiques d'accès mémoire

Si tous les noeuds travaillent sur le même espace mémoire, il peut se produire des problèmes d'accès si deux tâches tentent d'accéder au même emplacement.

3 Calcul distribué en réseau