

Cahier des charges : My Little Cluster

Cluster team

Table des matières

1	Présentation du groupe	2
1.1	Quentin de Laroussilhe	2
1.2	Camille Dollé	2
1.3	Guillaume Sanchez	2
1.4	François Boiteux	3
2	La parallélisation	3
2.1	Pourquoi paralléliser un calcul ?	3
2.2	Les contraintes de la parrallélisation	4
2.2.1	Les effets de bords	4
2.2.2	Les problématiques d'accès mémoire	4
3	Calcul distribué en réseau	4
3.1	Computer cluster	4
4	Etat de l'art	5
4.1	BOINC	5
4.2	Principe de fonctionnement	5

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 concrète pour commencer à appréhender toutes les contraintes qu'imposent les architectures parallélisées.



FIGURE 1 – Quentin de Laroussilhe

1.2 Camille Dollé

Bonjour ! Je suis très enthousiaste à propos du projet que nous allons tenter de réaliser lors de ce second semestre. Tout d'abord, c'est un projet à réaliser en C ! Ce projet me permettra donc d'apprendre à manipuler plus en détails les différents aspects du C. Ce projet comporte également un fort aspect réseau, or je suis une débutante en matière de réseau, contrairement aux autres membres du projet. J'aurai donc l'occasion d'apprendre une montagne de notions sur le réseau lors de ce projet. Le sujet que nous a proposé Underflow m'a tout de suite intéressée, le projet représente pour moi un vrai challenge car il y a une vraie difficulté à le réaliser. D'une part, car il y a beaucoup de connaissances à acquérir avant de les mettre en pratique et d'autre part car le réseau est un domaine (presque) inconnu pour moi.

Le projet représente donc pour moi une compétence de plus à acquérir en informatique bien que notre projet impose beaucoup de recherches et de travail.

1.3 Guillaume Sanchez

Cet energunene n'a qu'une seule idee en tete, biffer 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'intéressera à la parallélisation ! 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 vidéos, la musique et les soirées.

Je m'occuperais de la partie serveur du projet ce qui me permettra 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 importants. 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 conçus pour fonctionner sur des architectures parallélisées contiennent des effets de bord, c'est à dire que l'opération d'une instruction exécutée à un instant t dépend des instructions exécutées 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'exécuter leurs instructions. Cela revient strictement à une exécution séquentielle des instructions.

La première difficulté est donc de minimiser les effets de bords pour minimiser les temps d'attente pour exécuter 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

3.1 Computer cluster

Ce terme va jouer un grand rôle dans notre projet, c'est pourquoi nous allons définir maintenant ce qu'on nous désignons par celui-ci.

En anglais, le terme "cluster" signifie "grappe". "Computer cluster" désigne donc un regroupement de plusieurs ordinateurs indépendants. Chaque ordinateur sera qualifié de "noeud" et sera utilisé comme un serveur indépendant. Mais la "grappe" finale sera considérée comme une seule et même entité. Les buts d'une telle configuration sont multiples : dépasser les limitations d'un seul ordinateur, augmenter la disponibilité. . . En bref, le "computer cluster" sert à obtenir de plus grandes performances. En effet, la principale utilisation du "computer cluster" est le calcul parallèle. Comment regrouper plusieurs ordinateurs entre eux ? Le procédé est très simple : on relie les ordinateurs via le réseau, en général un réseau local rapide.

Dans notre cas, par exemple si nous voulions “cracker” un mot de passe, chaque noeud/ordinateur effectuera une tâche, par exemple ici, un noeud aura pour tâche d’essayer des combinaisons de mots de passe sur une certaine plage, et un autre s’occupera en même temps d’une autre plage.

4 Etat de l’art

Nous citerons ici la description d’un des projet de “Network Computing” appelé BOINC. Il existe toutefois d’autres projets de supercomputing dans le monde.

4.1 BOINC

BOINC est l’acronyme de Berkeley Open Infrastructure for Network Computing (Infrastructure non propriétaire de Berkeley dédiée au calcul en réseau). BOINC est une plate-forme de calcul partagé. C’est donc un programme qui permet de faire participer des volontaires à des projets scientifiques susceptibles de faire avancer la science dans différents domaines tels que la médecine, l’astronomie, la physique ou les mathématiques.

BOINC a une puissance totale moyenne de calcul d’environ 6,5 PFLOPS répartie sur environ 800 000 ordinateurs en septembre 2012 (pour comparaison, le supercalculateur le plus puissant à la même époque est le Sequoia d’IBM, qui atteint une puissance de 16,324 PFLOPS).

4.2 Principe de fonctionnement

La plate-forme BOINC suit un modèle simple : chaque projet s’étend sur un serveur central qui exécute une application Master. Les applications sont réparties sur des milliers de petites tâches qui sont envoyées à des machines réparties sur l’Internet, où ils exécutent des applications de type travailleurs. Il n’y a pas de communication entre les travailleurs. Toutes les communications doivent être du travailleur au Maître, afin de permettre la traversée de NAT et pare-feu. Les applications BOINC sont donc limitées au modèle Master/worker, avec un serveur central chargé de distribuer le travail aux clients BOINC. BOINC suit un protocole de réseau simple, qui demande aux clients d’initier toutes les communications (problèmes de NAT/pare-feu) et de communiquer avec le serveur à chaque fois qu’un client demande plus de travail. Chaque fois qu’un client est inactif et décide d’exécuter plus de travail, il doit contacter le serveur principal trois fois.

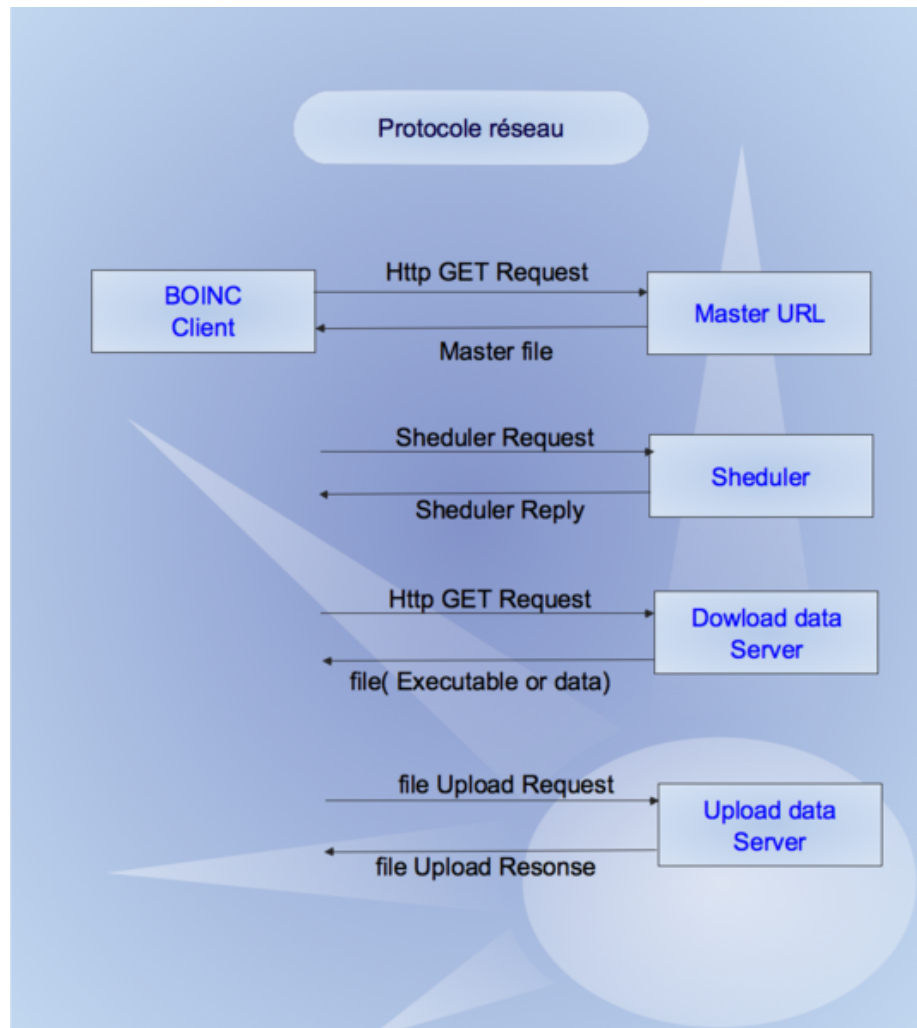


FIGURE 3 – Protocole réseau