TD4

Version du 7 mars 2022

1 Autour du projet

Exercice 1 – Algorithme de Lanczos par bloc

Le projet consiste à paralléliser l'algorithme de Lanczos par bloc. Il s'agit d'une méthode itérative pour résoudre des systèmes du type $xM = 0 \bmod p$. Ici N désigne la taille de la matrice M, qui est supposée creuse de densité d. Un *bloc de vecteurs de taille* n est simplement une matrice de taille $N \times n$ (n vecteurs colonnes).

Sans rentrer dans les détails, l'algorithme effectue N/n itération, et chaque itération fait :

- Deux produits matrice-vecteur $y \leftarrow Mx$ puis $z \leftarrow M^t y$, où x, y et z sont des blocs de vecteurs.
- Deux produits bloc-bloc du type $A \leftarrow z^t \times z$ et $B \leftarrow x^t \times z$, où A, B sont de petites matrices $n \times n$.
- Trois produits du type $x \leftarrow x + yC$ où x, y sont des blocs de vecteurs et C est une petite matrice $n \times n$.

Lorsque p est faible ($p = 2$ par exemple), on *doit* choisir des valeurs de n assez grandes, sinon le calcul risque d'échouer. Lorsque p est grand (comme dans le projet), on pourrait choisir n'importe quelle valeur de p.

1. Comment est-ce que le nombre d'opérations arithmétiques dépend de n ?
2. Comment est-ce que le volume de communications dépend de n ?
3. Comment a-t-on intérêt à choisir n ?

Exercice 2 – Calculs et communications

On s'intéresse au calcul de la séquence $x_{i+1} = Ax_i$, où A est une matrice creuse de taille n et de densité d, et x_0 un vecteur de taille n choisi de manière quelconque. C'est exactement ce qui se passe dans le projet. Il s'accompagne de défis à résoudre, qui sont résumés par le tableau ci-dessous.

matrix	n	d	dn	sur un coeur
easy	65 000	0.075%	50	1 heure
medium	200 000	0.0655%	130	24 jour
hard	700 000	0.036%	250	1 mois
HPC ! ! !	4 000 000	0.005%	200	5 ans

Cet exercice propose une reflexion sur les performances qu'on pourrait obtenir, en essayant de simplifier un peu les calculs.

Dans le cours, il est démontré qu'avec une répartition 1D de la matrice, calculer le prochain x_i prend un temps

$$T_1 = \frac{2dn^2}{pC} + \frac{n}{D}$$

Le premier terme correspond au temps de calcul, le deuxième à la durée des communications. Ici, C désigne la puissance de calcul d'un serveur de calcul (en FLOPS) et D désigne la bande-passante du réseau (en coefficients de vecteur par seconde). Un *cluster* typique de grid'5000 a $C \approx 10^{10}$ FLOPS et $D \approx 10^8$ (il ne s'agit que d'un ordre de grandeur).

Dans le cours, il est également démontré qu'avec une répartition 2D de la matrice, calculer le prochain x_i prend un temps

$$T_2 = \frac{2dn^2}{pC} + \frac{n}{D\sqrt{p}}$$

Les deux tendent vers zéro lorsque p augmente, ce qui est nettement mieux. Mais cette remarque n'épuise pas la question car les deux n'évoluent pas à la même vitesse.

Pour simplifier, on ignore le plus petit terme des deux. Si le temps de calcul domine la durée des communications, on ignore ces dernières, et vice-versa. Il y a donc un *seuil* où calcul et communications s'équilibrent lorsque p varie.

1. Discuter de l'accélération qu'on peut espérer obtenir avec une distribution 1D de la matrice, en deçà et au-delà du seuil.
2. Discuter de la résolution des défis du projet avec une répartition 1D de la matrice.
3. Refaire le même raisonnement avec une répartition 2D de la matrice. Quel est le seuil, et que se passe-t-il en-deçà / au-delà ?
4. Discuter de la résolution des défis, notamment des deux plus durs, avec une répartition 2D de la matrice.

2 Réseaux pour le HPC

L'hypothèse, affirmée en cours, que transmettre un message de n octets prend un temps $\alpha + \beta n$, indépendamment du noeud de départ, du noeud de destination, et de ce qui se passe ailleurs, n'est pas forcément très réaliste. Les deux exercices suivant montrent ce que cela implique.

Exercice 3 – Coût caché du réseau

L'hypothèse implique que, pour toute permutation σ des processeurs, le processeur i peut envoyer un message au processeur $\sigma(i)$ sans congestion ni délai. Cet exercice montre qu'un réseau qui permet ceci coûte cher.

Supposons une machine à n noeuds en forme de cube. On suppose que les processeurs sont disposés régulièrement dans l'espace à trois dimension. Autrement dit, s'il y a un processeur de coordonnées (x, y, z), alors il y en a d'autres en $(x \pm 1, y, z)$, $(x, y \pm 1, z)$ et $(x, y, z \pm 1)$, à condition que ces coordonnées soient dans le cube.

La longueur totale des cables est notée $\alpha \cdot n^{1+z}$, pour un certain z qu'on voudrait minorer.

1. On suppose que les cables occupent un volume proportionnel à leur longueur. Donner une borne inférieure sur la taille du cube.
2. On coupe la machine en trois parties égales selon l'un des axes. Donner une borne inférieure sur le nombre de cables qui doivent connecter le tiers gauche au tiers droit pour que les communications puissent se faire sans congestion.
3. Donner une borne inférieure sur longueur cumulée de ces cables.
4. En déduire une minoration de la longueur totale des cables dans la machine.
5. Que peut-on conclure sur le coût d'un réseau capable de supporter l'hypothèse ci-dessus ?

Exercice 4 – Réseau en tore 2D

On considère un réseau de p serveurs de calculs disposés sur une grille à deux dimensions (supposons qu'elle est de dimension $\sqrt{p} \times \sqrt{p}$). Chaque noeud est directement relié à ses 6 voisins. Cette fois, la longueur des cables est *proportionnelle* au nombre de processeurs.

On suppose que transmettre un message de n octets le long d'un cable prend un temps $\alpha + n\beta$, et *cette fois* c'est réaliste.

1. Comment former un tore 2D (*donut*) arbitrairement grand avec des cables réseaux de taille fixe ?
2. Combien de temps faut-il, dans le pire des cas, pour transmettre un message d'un noeud à un autre ?
3. Les lignes $x = n/4$, $x = n/2$ et $x = 3n/4$ « coupent » la machine en quatre quarts. Supposons que chaque noeuds du quart numéro un échange un message avec un noeud quelconque du quart numéro 3. Donner une borne inférieure sur la durée d'acheminement des messages.
4. Décrivez un algorithme pour effectuer le `Broadcast` (de taille n) dans ce réseau. Combien de temps cela prend-il ?
5. Décrivez un algorithme pour effectuer le `AllGather` de taille n dans ce réseau. Combien de temps cela prend-il ?