TP Noté - Calculs parallèles

Created	@3 juin 2024 14:26
Type	Compte Rendu
Reviewed	☐

> CHAIX Maxence, MARI Farid, ROTH Tristan, BOURDON Marin — AIL2

Git : https://github.com/tristan-roth/Projet_raytracing

Préambule

Questions :

En utilisant votre meilleur outil, votre imagination, décrivez et illustrez comment cela pourrait être réalisé, sans rentrer dans les détails JAVA, que vous n'allez pas tarder à mettre en œuvre.

Pour réduire le temps de calcul, nous allons diviser l'image en plusieurs fragments et charger chaque partie indépendamment. Une fois le traitement de chaque fragment terminé, nous les rassemblerons pour reconstituer l'image complète. Cette approche permet d'exploiter le calcul parallèle et d'accélérer ainsi le processus global.

Le tracé de rayon (*raytracing*)

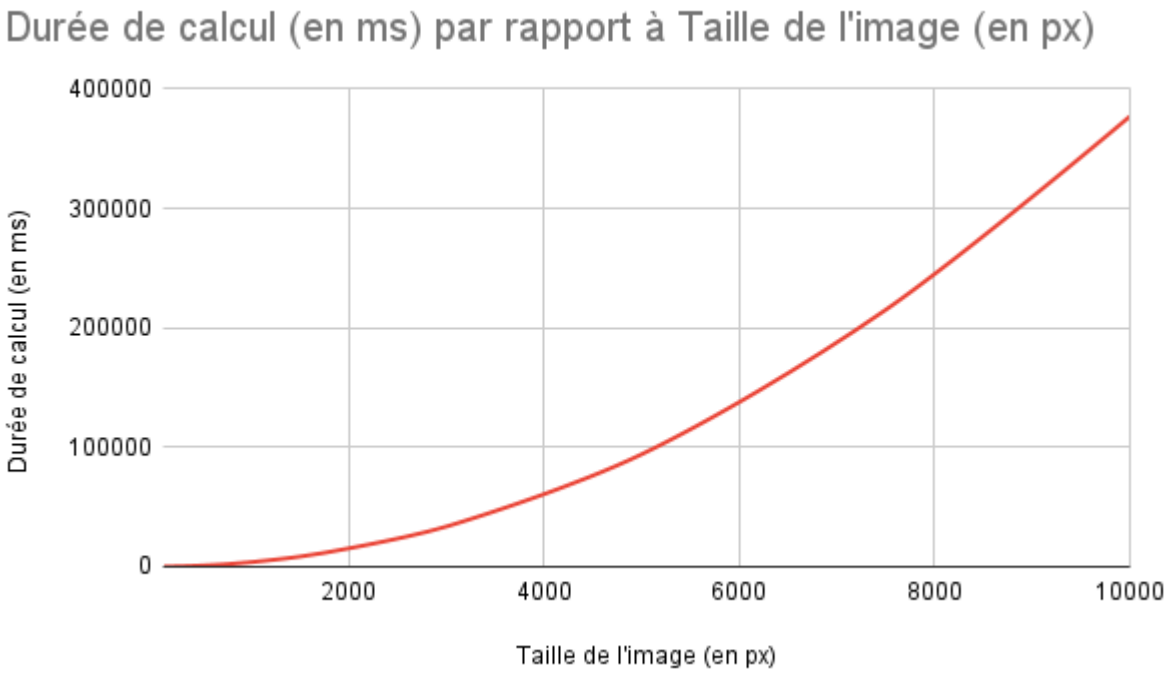
Questions :

1. Tester le programme en modifiant ses paramètres (sur la ligne de commande)

2. Observer le temps de d'exécution en fonction de la taille de l'image calculée. Vous pouvez faire une courbe (temps de calcul et tailles d'image).

Nous avons mesuré le temps de calcul sur un MacBook Air équipé d'un processeur M2 8 cœurs. Nous avons également décidé de faire varier la taille de l'image tout en gardant le ratio carré entre 100 et 10 000 pixels.

Taille de l'image (en px)	Temps de calcul (en ms)
100	84
150	140
200	211
300	382
500	956
750	2060
1000	3783
1500	8439
2000	15310
3000	33623
5000	94114
7500	215097
10000	377029



3. En ne modifiant que le fichier *LancerRaytracer.java*, reproduire l'image suivante

```
Image image = scene.compute(x0, y0, l, h);

...

// Affichage de l'image calculée
disp.setImage(image, x0, y0);
```

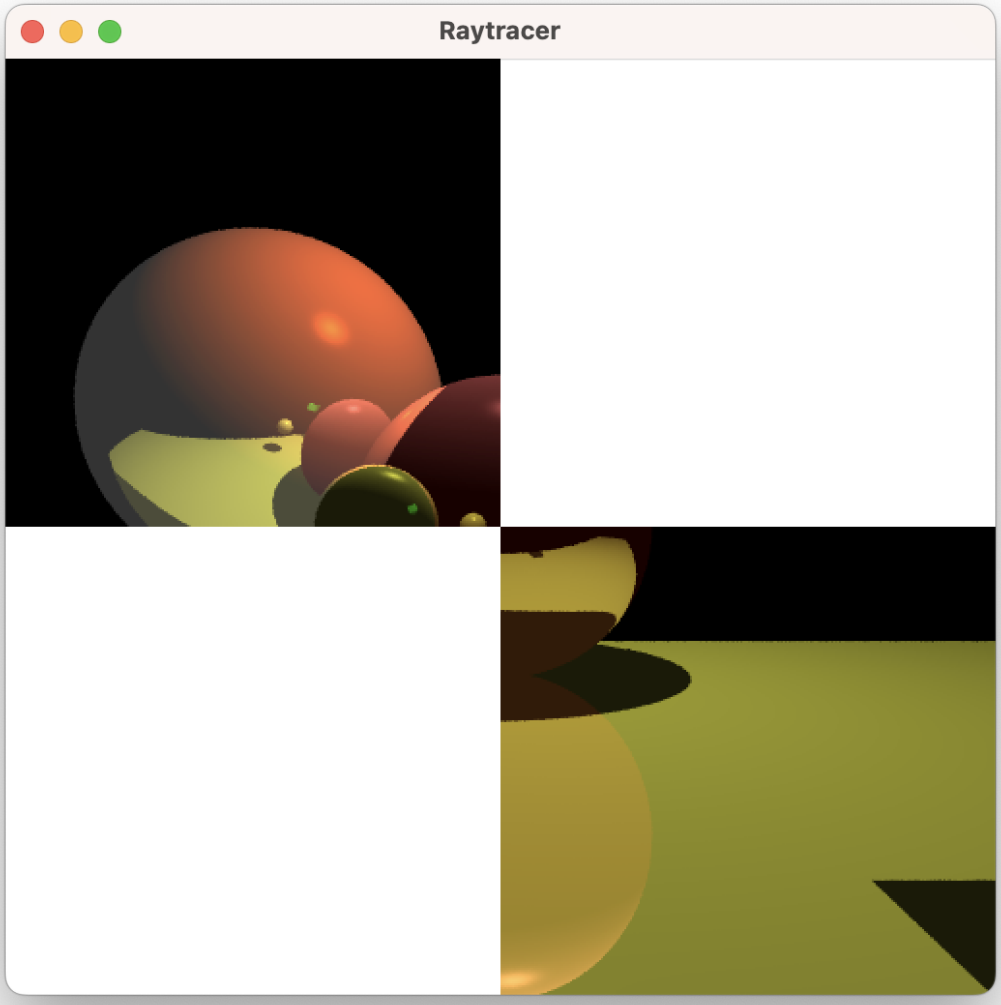
Dans le code actuel, la ligne `Image image = scene.compute(x0, y0, l, h);` calcul l'image complète, et la ligne `disp.setImage(image, x0, y0)` affiche l'image dans la fenêtre.

Pour reproduire l'image, il suffit de générer deux images (une pour le coin en haut à gauche, et une pour le coin en bas à droite) et des les ajouter toutes les deux dans la fenêtre :

```
Image hautGauche = scene.compute(x0, y0, l/2, h/2);
Image basDroit = scene.compute(x0+l/2, y0+h/2, l/2, h/2);

...
```

```
// Affichage des images calculées
disp.setImage(hautGauche, x0, y0);
disp.setImage(basDroit, x0+l/2, y0+h/2);
```



Accélérons les choses

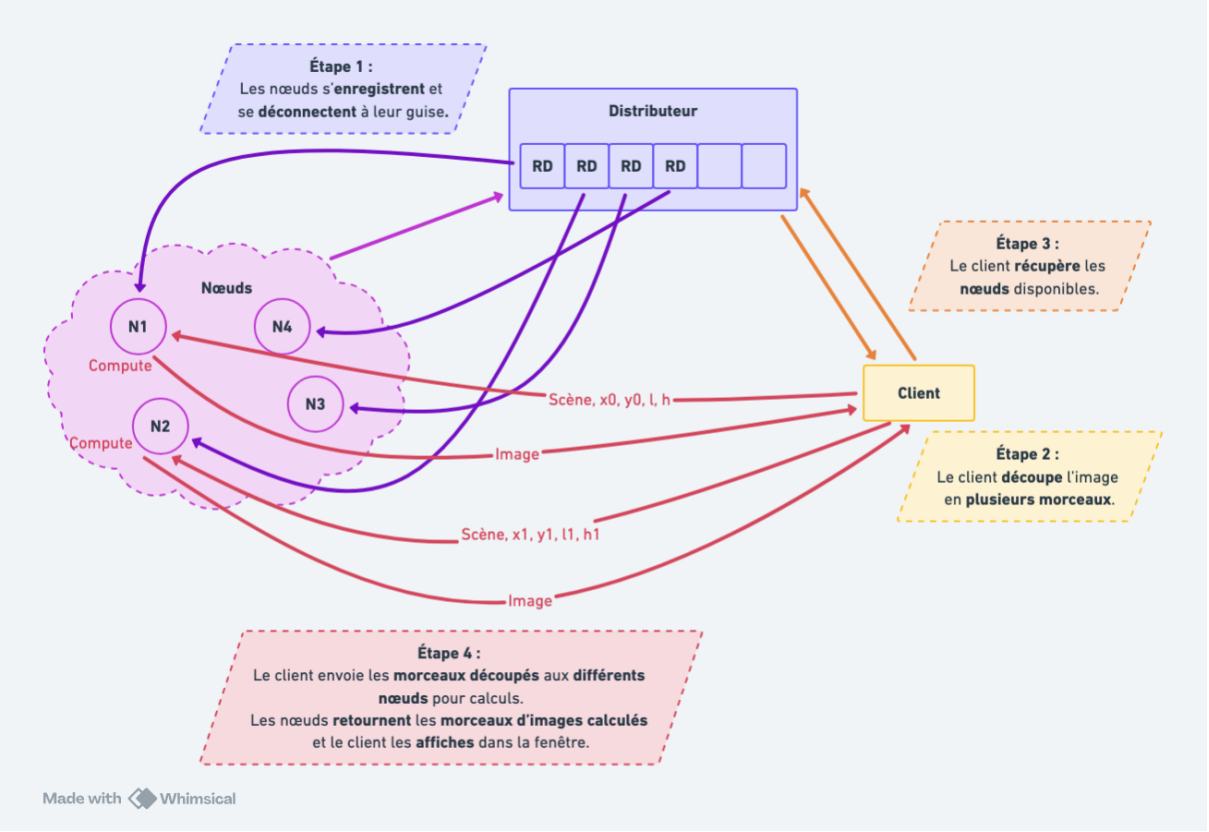
Questions :

Faire un petit schéma de cette architecture en identifiant les choses suivantes :

1. **Le/les processus fixes (ceux qui écoutent sur un port**

choisi) et les processus éphémères ? (ceux qui rentrent et sortent a leur guise) ?

2. Les types de données échangées entre les processus



Le `Distributeur` est un processus fixe, tandis que les `Noeuds` sont des processus éphémères.

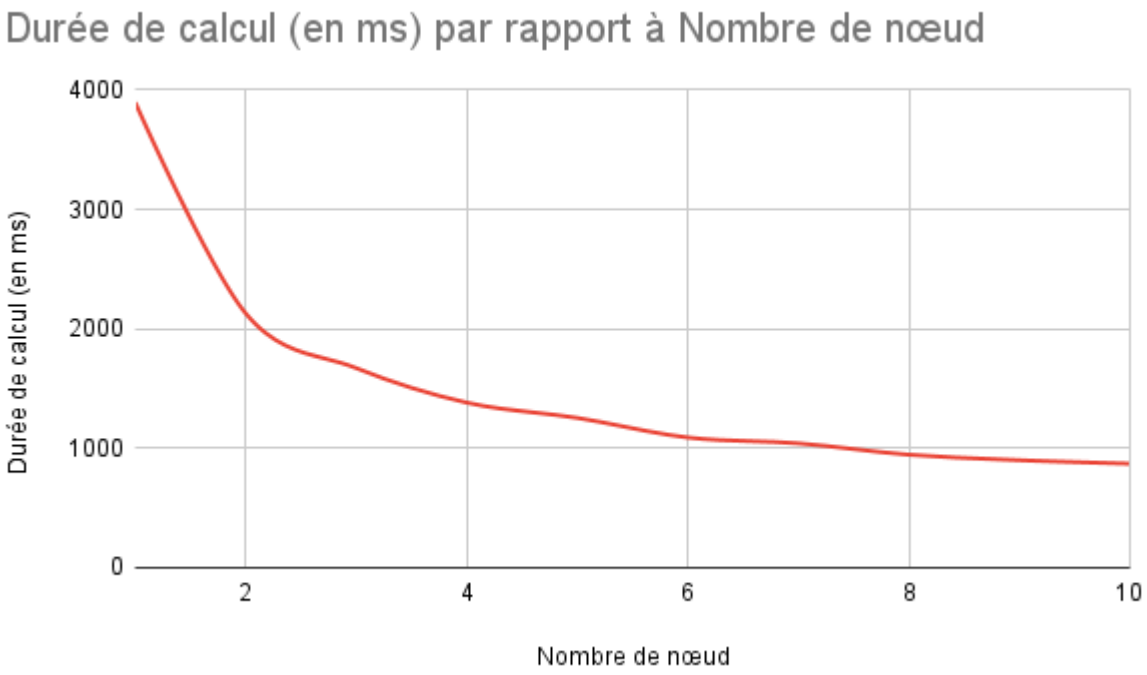
Si on veux que les calculs se fassent en parallèle que faut-il faire ?

Pour que les calculs se fassent en parallèle, nous devons utiliser des **Threads**. Dans chaque **Noeud**, nous lancerons un **Thread**.

Lancez-vous et réalisez cette application répartie ! Vérifiez que le calcul est bien accéléré.

Pour vérifier que le calcul est bien accéléré, nous avons décidé de relevé la durée de calcul selon le nombre de nœud disponible. Ce test a été réalisé sur un même et unique ordinateur (Macbook Air M2 8 cœur), les nœuds sont donc sur le même ordinateur.

Nombre de nœud	Durée de calcul (en ms)
1	3896
2	2129
3	1671
4	1382
5	1254
6	1090
7	1041
8	946
9	900
10	871



Nous pouvons voir que la durée de calcul réduit lorsque le nombre de nœud augmente, mais la durée commence à stagner lorsque l'on approche des 8 nœuds disponibles.