Calcul parallèle en finance
2019/2020

Projet de Calcul Parallèle

Lebihan Lucas - Couté Lucas
3ème année IF
Grenoble INP – ENSIMAG

Mars 2020

1 Introduction

Le but de ce projet est de paralléliser le pricer développé en début d'année afin d'améliorer les performances en temps de ce dernier. Nous allons pour cela utiliser la bibliothèque de parallélisation OpenMPI.

2 Question 1

Voici le schéma des appels de fonctions lors de l'exécution de notre pricer.

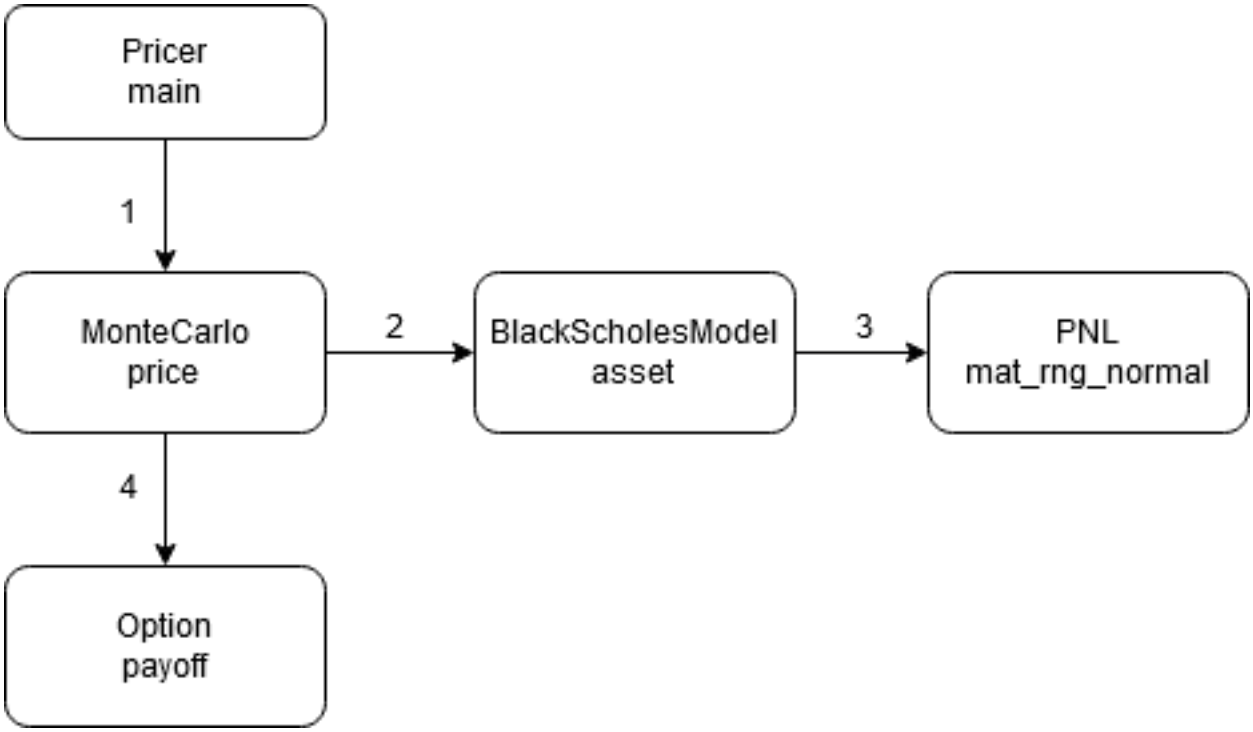

Figure 1: Schéma d'appels fonctions pricer itératif

Nous pouvons voir ici que la classe MonteCarlo appelle la méthode *price* (1). Celle-ci pourra être parallélisé, car elle boucle plusieurs fois le même appel à la méthode *asset* de BSM (2) et chaque appel contribue à la modification de deux mêmes variables.
Cette boucle s'apprête parfaitement au principe maître-esclave de la parallélisation avec OpenMPI. En effet, un processus maître lancera donc la boucle et les autres processus esclaves, qui effectuerons chacun un certain nombre de tours de boucle en parallèle et retournerons chaque résultat au processus maître au travers des appels à MPI_Gather.
Attention cependant, nous pouvons voir dans ce schéma que chaque appel à asset lance un appel à *pnl_mat_rng_normal* de la librairie PNL (3). Il faudra donc aussi paralléliser la génération des nombres aléatoires afin que chaque processus n'utilise pas les mêmes nombres aléatoires. Nous utiliserons donc pour ça la parallélisation fournie par la bibliothèque

PNL avec des générateurs indépendants avec des id différents (pnl_rng_dcmt_create_id).

3 Question 2

Voici le nouveau schéma des appels de fonctions lors de l'exécution de notre pricer après la parallélisation à l'aide d'OpenMPI.

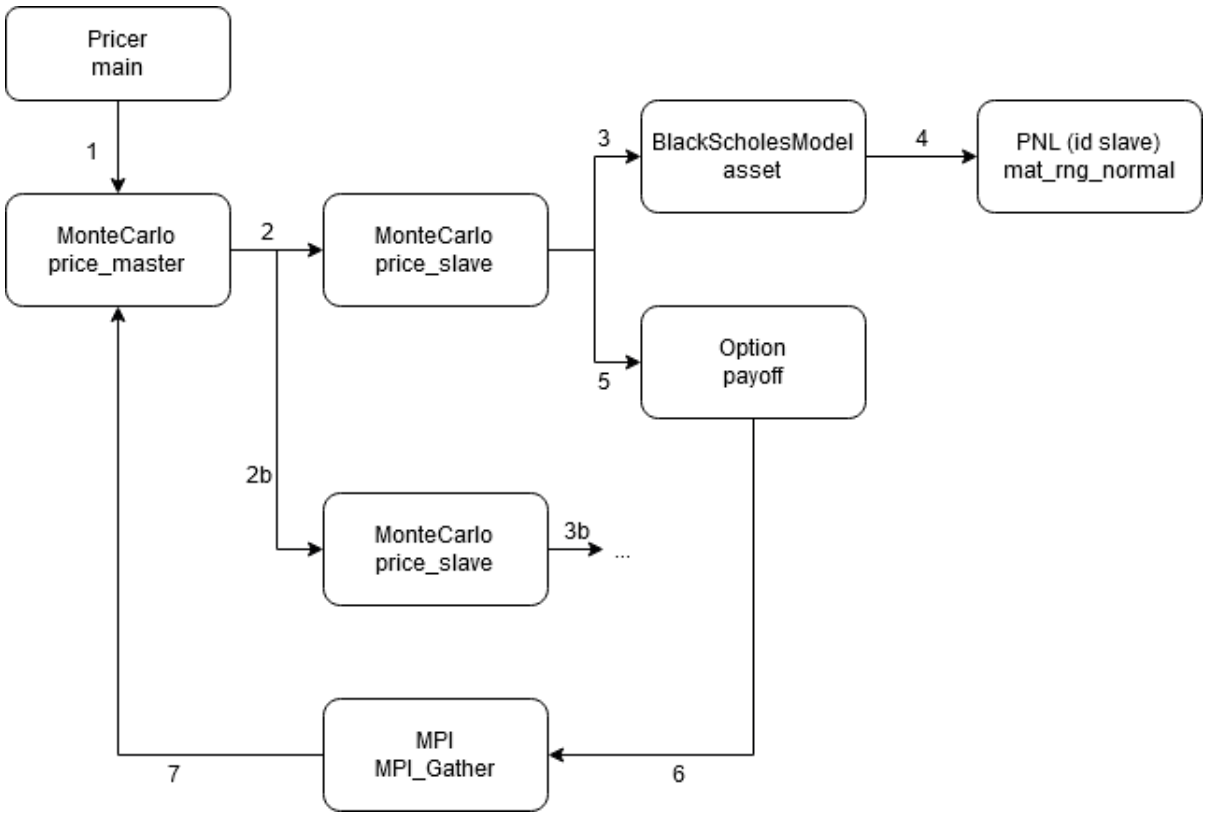

Figure 2: Schéma d'appels fonctions pricer parallèle

4 Question 3

La génération des nombres aléatoires se fera avec la biliothèque PNL. En effet, chaque processus créera son propre générateur indépendant grâce à la méthode *pnl_rng_dmct_create_id* qui prendra en argument le rang du processus.

5 Question 4

Voir code pour l'implémentation.
Voici nos résultats :

	basket	basket_1	basket_2	asian	perf
Prix parallèle	13.6569	13.6342	9.21003	4.75614	1.25705
Prix itératif	13.5988	13.6403	9.23947	4.71605	1.2575
IC parallèle	0.0250014	0.0251455	0.055509	0.0300923	5.9e-4
IC itératif	0.0251643	0.0250984	0.0557858	0.0300414	5.8e-4
Temps parallèle	0.28011	0.283419	0.298027	0.851111	0.280677
Temps itératif	0.48	0.49	0.48	2.32	0.47

De ces résultats nous en déduisons donc les deux mesures de performances :

	basket	basket_1	basket_2	asian	perf
Speedup	1.7136	1.7289	1.6106	2.7258	1.6745
Efficacité	0.4284	0.4322	0.4026	0.6814	0.4186

L'efficacité aurait pu donc être amélioré avec une parallélisation plus fine du code mais que nous n'avons pas réalisé faute d'idées. Il faut aussi remarquer que seulement 3 processus esclaves effectuaient les tours de boucles Monte Carlo. Le processus maitre ne servait qu'à récupérer les données envoyées par les processus esclaves.

6 Question 5

Voir code pour l'implémentation.
Pour s'assurer d'avoir la bonne précision, il faudrait aggréger les résultats à chaque tour de boucle d'un des processus esclave et vérifier la précision actuelle. Cependant, cela n'est que très peu efficace car cela introduit un très grand nombre de communications MPI (2 pour chaque tour de boucle de chaque processus) tout en redevenant un algorithme plus ou moins itératif (le processus maître récupère les données à chaque tour de boucle de chaque processus).
Nous avons décidé d'aggréger les résultats des processus slaves sur le processus master tous les 1000 tirages. Si la précision souhaitée n'est pas atteinte, on recommence 1000 tirages. Ainsi de suite jusqu'à atteindre la précision souhaitée et retourner les résultats.
Comme les communications entre les processus prennent du temps, nous avons essayé de les minimiser, il fallait donc trouver un compromis avec le nombre de tirages par tour de boucles car le temps de calcul augmente avec ce dernier mais le temps de communication diminue.
Après plusieurs essais, nous sommes donc restés avec 2500 tirages avant chaque vérification de la précision.