Compte Rendu Projet BlackJack

CABOUAT Charlotte
SANTORO Fabrizio

Avril 2018

1 Implémentation

La Simulation

Pour la simulation, nous avons choisis d'utiliser les pthread. La mémoire partagée nous semblait la plus pertinente pour le problème, ainsi que la gestion des pthreads.
Les joueurs sont donc des threads créés par la banque (le programme principal), qui communique avec la banque à travers leurs mémoire partagé et leurs barrières.

2 Difficultés

Log des Joueurs

A chaque fin de manche, les joueurs doivent écrire dans un fichier un compte-rendu de leurs manche.
La difficulté principale rencontrée a été la conversion des entiers en caractères et vice-versa.
Pour utiliser au maximum le parallélisme, chaque joueur écrit dans son propre fichier.
Bien que la vitesse l'accès au disque soit limité et que le log ne soit pas écrit en bloc(batch), nous avons une amélioration légère.

Les barrières

Difficulté dans la gestion des barrières ; le nombre de barrières à utiliser, le nombres de joueurs à attendre dans la barrière.
Communication entre barrière banque et barrière joueur, le nombre de joueurs dans la barrière peut être modifié, il fallait donc réinitialiser la barrière à chaque début de round, d'où la création de la fonction getNbPlayerPlay.
Donc la barrière pouvait changer d'adresse, c'est pour cela que nous avons fait un pointeur de pointeur, ainsi que la nécessité des barrières temporaires.

En effet, les barrières ne pouvant pas être mise à jour pendant qu'elles sont utilisées, une barrière temporaire est utilisée.

La Synchronisation entre les joueurs et la banque. La banque doit pouvoir communiquer avec tous les joueurs, savoir si ils veulent une carte ou non, connaître le nombre de joueurs restant pour le round.
Gérer les différents cas de gain, toujours prendre en compte les résultats du dernier round, pour connaître les gains du round actuel. Création de flags. Comprendre exactement le déroulement d'un round.

3 Structure de la Main

Dans un souci d'optimisation de la mémoire, nous avons opté pour une structure de données basée sur les listes chaînées. La structure est donc composée d'un tableau de 2 entiers (cardvalue_t) et un pointeur vers la prochaine structure, pour un total de 16 byte.
En observant les probabilités de tirer plusieurs cartes, nous pouvons voir que dans un scénario à 4 cartes en main, la structure personnalisée utilise 16 + 16 byte, alors qu'une simple liste chaîné $(4 + 8) \cdot 4$.
Nous pouvons aussi remarquer que du fait que la taille de la structure soit 16, elle pourrait rentrer dans une ligne de cache/vectorisée plus facilement.

4 La Banque

La banque est le processus principal, elle est chargée de distribuer les cartes à chaque début de round puis à la demande des joueurs. Elle sait aussi le nombre de joueurs dans la partie et vérifie les gagnants et les gains des joueurs. C'est elle qui met à jour les barrières et s'occupe de la coordination de la simulation. Pendant la fin du tour de main, elle s'occupe de trouver le vainqueur (si il y en a un), en gérant les cas spécifiques.

5 Les Joueurs

Les joueurs sont des threads créés par le processus principal. Ils peuvent demander des cartes à la banques tant qu'ils ne dépassent pas leurs valeur maximale, ici stopVal, indiqué dans la configurations des joueurs. La stratégie du joueur est préconfigurée à partir du fichier spécifié comme paramètre du programme, à travers un flag, le joueur décide de sa mise à chaque début de tour de main.