Ingénierie des modèles & Simulation – Métaprogrammation – ZZ3 F2 TP2

But du TP: Tester une technique de génération de code par un autre programme – on parle de méta-programmation. Application simple : simulation de Monte Carlo avec pré-génération des valeurs pseudo-aléatoires. Le début du TP repart des acquis en 2nd année (introduction à la simulation aléatoire à événements discrets).

1. Retrouver la dernière version du code source en C du générateur « original » Mersenne Twister de Makoto Mastumoto sur sa « Home page » (nommé MT ci-après). Aller dans ses anciennes pages (avant 2010) jusqu'à trouver (MT – Explanations & C code – la dernière version plus sure et efficace publiée en 2002 (dernière implémentation en mars 2004). Compiler ce code et comme en 2nd année (re-) **tester que le programme est bien porté sur la machine que vous utilisez actuellemennt.** Les sorties doivent être conformes à ce qui est annoncé par l'auteur qui donne un exemple du résultat attendu (fichier .out). Le README de Matsumoto explique que le fichier '.out' contient les résultats que l'on doit obtenir pour montrer que le générateur a été bien porté et que l'on reproduit bien la même trace. Ici le principe de portabilité est lié à la répétabilité et à la reproductibilité scientifique.

2. Utiliser ce générateur pour produire un code source en C qui contient simplement une déclaration et une initialisation de tableau avec N nombres pseudo-aléatoires tirés suivant le générateur MT (le code boucle pour produire un nombre par ligne (puis une virgule) et on termine avec ' } ; '. Un programme va donc écrire un fichier source '.c' qui servira pour un autre programme.

```
double tabMT[] = {
   premier nombre tiré,
   deuxième nombre,
   etc.,
   …
   } ;
```

3. Reprendre un code de simulation de Monte Carlo pour calculer PI (cf. ZZ2 – ou à retrouver sur le Net ou encore à reprogrammer rapidement). Tester ce code en faisant des appels au générateur MT en faisant 10 réplications séquentielles avec 1 000 000 de points (10^7 points soit 2 x 10^7 tirages au total). Mesurer le temps de calcul (avec le time Unix) et donner la moyenne et l'écart moyen avec la constante M_PI.

4. Reprendre le code du point 3 et intégrer le code généré au point 2 pour 10 millions de points (2 x 10^7 tirages). Utiliser les nombres stockés dans un tableau plutôt que de lancer les tirages en appelant le générateur (optimisation dite par « lookup table »). La compilation séparée du code qui déclare le tableau est conseillée (accès avec « `extern double tabMT[2 * 10 000 000];` »). Vous pouvez néanmoins tout mettre dans un même fichier source peu simple à manipuler… Noter le temps de compilation et d'édition des liens. Attention : le tirage chaque point nécessite 2 nombres pseudo-aléatoires. La zone de swap doit être assez grande pour cette étape de compilation.

5. Comparer les performances en temps machine, sur un PC et le serveur, entre : (1) entre tirer des nombres pseudo-aléatoires avec MT et (2) lire ces nombres pré-calculés dans le tableau préalablement généré. Prendre un nombre significatif de tirages en fonction des limites de la taille possible pour une compilation et la zone de swap spécifique pour la compilation (qui pourrait être étendue sur un PC).

6. Ajouter plusieurs ordres de grandeur et mesurer également le temps de calcul (pour des simulations utilisant 2.10^9 points) en effectuant réellement les tirages. Pourrez-vous utiliser la technique du tableau pré-calculé et stocké pour ces milliards de points ? Quels seraient les problèmes / contraintes en temps et en espace si l'on souhaite utiliser une technique de méta programmation pour la question (générant 2 x 10 milliards de nombres) ?

7. Pour information une technique logicielle permet d'utiliser au mieux le matériel. Au lieu de générer un code source avec un tableau de nombres issus de MT, on peut écrire ces nombres dans un fichier (binaire) et (re)découvrir la technique dite du « memory mapping » pour adresser le fichier binaire comme s'il s'agissait d'un tableau en mémoire (https://linux.die.net/man/3/mmap). Cette technique est opérationnelle sur de nombreux systèmes et les fichiers peuvent être placés sur des disques virtuels qui sont en mémoire afin d'obtenir de très bonnes performances.