Catch me if you can !

Marion Dalle, Jean-Marc Vincent, Florence Perronnin *

Univ. Grenoble Alpes
F-38000 Grenoble, France
CEA, LETI, MINATEC Campus
F-38054 Grenoble, France
marion.dalle@cea.fr

1. Introduction

L'évaluation des performances de très grands réseaux et systèmes nécessite parfois une simulation très longue de modèles markoviens, que ce soit par exemple pour capturer des séquences d'événements rares, ou bien pour évaluer des taux de couverture, ou encore pour analyser des basculements de comportement dans des modèles "raides".

Pour réduire la durée de la simulation d'une seule trajectoire longue, l'idée est d'utiliser plus efficacement les ressources à disposition, en particulier les ressources de calcul (cœurs), en anticipant les calculs sur des tranches de temps se recouvrant au minimum. L'argument clé est, sous certaines conditions, l'efficacité du couplage de trajectoires qui permet de garantir l'obtention de la trajectoire complète.

L'objectif de ce travail est donc de proposer un algorithme parallèle de simulation de trajectoires longues basé sur une technique de "couplage horizontal" de trajectoires, d'analyser son coût et de donner les premiers résultats d'efficacité.

2. Simulation de modèles markoviens

On considère un modèle markovien à temps discret $\{X_t\}_{t\in\mathbb{N}}$ décrit par sa fonction de transition $\Phi : X_t = \Phi(X_{t-1}, e_t)$ où e_t est l'événement se produisant à l'instant t (généré aléatoirement). La simulation classique d'une trajectoire de longueur T consiste, pour chaque instant, à générer aléatoirement l'événement selon les probabilités du modèle puis à mettre à jour l'état selon la fonction de transition (algorithme 1).

Algorithme 1 : Algorithme séquentiel

Données : x_0 (état initial), T
pour t *de 1 à* T **faire**
  $e_t \leftarrow genere\_evenement()$;
  $x_t \leftarrow \Phi(x_{t-1}, e_t)$;

Une technique de parallélisation consiste à calculer séparément des segments disjoints de la trajectoire, en anticipant l'état de départ dans chacun de ces segments. Cette technique initialement proposée par Nicol et al. [4] ou Fujimoto [3] est la suivante :

La période simulée est partitionnée en P *tranches* de temps de durée fixe $\Delta = T/P$ démarrant à $t_i = i\Delta$ pour $0 \leqslant i \leqslant P-1$ affectées respectivement aux P cœurs. Pour chaque cœur on génère également la séquence des événements et l'état initial à l'instant t_i. Plusieurs *phases* s'ensuivent durant lesquelles : chaque cœur i simule la trajectoire du segment $[t_i, t_{i+1}]$ selon l'algorithme 1 en partant de l'état x_{t_i} ; les états de début de tranche sont corrigés par la valeur obtenue en fin de la tranche précédente à la fin de la phase précédente. Les phases sont répétées jusqu'à ce que les états de début de tranche ne soient plus modifiés.

Il est clair que dans le pire des cas cet algorithme simulation parallèle nécessite autant d'étapes de calcul que l'algorithme séquentiel (au pire P phases).

Le temps de calcul peut encore être réduit au prix de la précision en simulant un *ensemble d'états possibles* au lieu d'une trajectoire exacte [2]. L'idée est de remplacer le calcul de l'état par le calcul de bornes sur cet état : ainsi, les conditions aux limites des tranches de temps sont moins strictes (inclusion) ce qui réduit le nombre de phases de l'algorithme. Cela permet d'obtenir rapidement une estimation de la partie de l'espace d'état atteinte par le modèle.

3. L'algorithme *Catch me if you can*

L'algorithme que nous proposons vise à adapter la méthode développée par Nicol et al. en conditionnant l'arrêt du calcul de la trajectoire sur une tranche de temps par une condition de couplage.

3.1. Couplage de chaînes de Markov

Considérons une séquence d'événements $\{e_1, e_2, \cdots, e_T\}$ qui permet la génération de deux trajectoires $\{X_t^a\}$ et $\{X_t^b\}$ par applications successives de Φ à partir des états initiaux x_0^a et x_0^b. Étant donné la nature du schéma itératif, si les deux trajectoires couplent à l'instant t alors elles seront confondues après t. Le premier instant de rencontre des deux trajectoires est appelé instant de couplage, noté τ.

3.2. Principe de parallélisation

On suppose que la séquence des événements est fixée et partagée en lecture par tous les cœurs. Elle peut être calculée au préalable ou à la volée en parallèle. À

*. Ce travail a été partiellement financé par l'ANR Marmote

chaque cœur i, $0 \leqslant i \leqslant P-1$, on attribue un intervalle de temps de taille variable pour calculer un fragment de trajectoire. C'est à dire une date de départ et un état de départ x_{t_i} correspondant (fixé arbitrairement ou par une heuristique).
Chaque cœur i calcule donc en parallèle la trajectoire issue de x_{t_i} à partir de la date t_i. Le calcul est arrêté dès que la trajectoire couple avec une trajectoire déjà calculée par un autre cœur, ce qui permet d'éviter des calculs redondants. C'est pourquoi cet algorithme s'appelle *catch me if you can*. L'algorithme proposé est illustré en Figure 1.

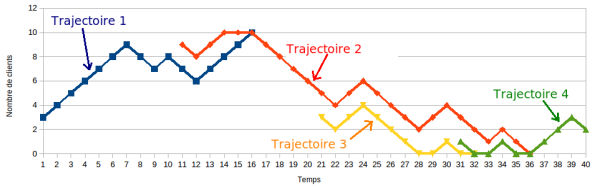

FIGURE 1 – Simulation d'une période de 40 itérations sur 4 cœurs. La durée totale de simulation est 25 (temps de couplage entre trajectoires 2 et 4) au lieu de 39 en séquentiel.

Si le principe de cet algorithme est simple, les difficultés résident dans sa mise en œuvre. En effet, même si l'on travaille en mémoire partagée, le test compare des données associées à des cœurs différents, ce qui correspond à des synchronisations en mémoire qui peuvent dégrader fortement les performances de l'algorithme.
Du point de vue théorique, si on note N_i le nombre d'itérations effectuées par le cœur i avant couplage, la durée totale de simulation (*makespan*) en nombre d'itérations est $N = \max_i N_i$. Pour des $t_{i+1} - t_i$ de l'ordre de $\frac{T}{P}$ on peut montrer que :

$$N \simeq \frac{T}{P} + \alpha,$$

où α correspond au surcoût lié au couplage de trajectoires et peut être majoré stochastiquement par le maximum de P temps de couplage, ce qui croît très lentement en fonction de P et laisse espérer au moins une efficacité asymptotique.

4. Premières analyses

L'un des critères de performance de l'algorithme est le nombre d'itérations (calculs de la fonction Φ) nécessaires à l'obtention de la trajectoire complète.

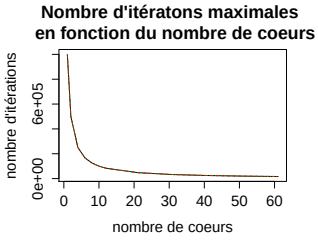

FIGURE 2 – Modèle de file M/M/1/50 charge $\rho = 80\%$, période de simulation $T = 100000$. 300 répétitions d'expériences par point. Intervalles de confiance inférieurs à 1%

L'exemple de la figure 2 illustre l'accélération obtenue en parallélisant le calcul de la trajectoire. Dans ce modèle le temps moyen de couplage est grossièrement borné par une fonction linéaire en la capacité de la file [1] (ici de l'ordre de 250 itérations), ce qui est négligeable devant $\frac{T}{P}$. Par contre si l'on augmente le nombre de cœurs le temps de couplage devient non-négligeable. Cet effet sera accentué par le coût lié aux contraintes de synchronisations entre les différents cœurs. Dans notre cas de simulation un nombre de cœurs de l'ordre de 10 donnerait un bon compromis entre accélération et utilisation de ressources.

Bibliographie

1. Jantien Dopper, Bruno Gaujal, and Jean-Marc Vincent. Bounds for the Coupling Time in Queueing Networks Perfect Simulation. In *Numerical Solutions for Markov Chain (NSMC06)*, pages 117–136, Charleston, June 2006.
2. Jean-Michel Fourneau and Franck Quessette. Monotonicity and efficient computation of bounds with time parallel simulation. In *Computer Performance Engineering*, volume 6977 of *LNCS*, pages 57–71. Springer Berlin Heidelberg, 2011.
3. Richard M. Fujimoto. *Parallel and Distributed Simulation Systems*. Wiley-Interscience, 2000.
4. David Nicol, Albert Greenberg, and Boris Lubachevsky. Massively parallel algorithms for trace-driven cache simulations. *IEEE Trans. Parallel Distrib. Syst.*, 5(8) :849–859, 1994.