Chapitre 3

Étude de cas : modélisation d'algorithmes de gestion de mémoire paginée

L'objectif de ce chapitre est de présenter un problème de modélisation d'algorithmes de gestion de mémoire paginée. L'accent sera plus précisément porté sur la méthodologie de modélisation de tels algorithmes et sur interprétation des résultats mathématiques obtenus par le traitement des modèles.

3.1 Le problème

La mémoire d'un ordinateur est structurée en un ensemble de pages mémoires. Un programme, s'exécutant sur le processeur, appelle successivement les pages mémoire dont il a besoin, effectue le traitement associé à ces pages et replace la page traitée dans la mémoire. Pour l'utilisateur, la gestion des pages est entièrement transparente, la programmation se fait sans contrainte d'usage de la mémoire. C'est pour cette raison qu'une telle structure de mémoire est appelée mémoire virtuelle, tout se passe comme si l'espace mémoire utilisable était infini.

L'architecture matérielle pour réaliser une telle fonctionnalité est présentée en figure 3.1, pour plus de précisions sur de tels systèmes informatiques on consultera les ouvrages [26] et [43].

Le système d'exploitation est alors chargé de la gestion et de l'organisation de la mémoire. Comme la mémoire est localisée sur 2 sites physiques différents (la mémoire du processeur et le disque de pagination), le temps d'accès à une page donnée dépend de la localisation de la page. Dans la plupart des ordinateurs, le temps d'accès à une page sauvegardée sur le disque est bien plus grand que le temps d'accès à une page située en mémoire. Le système d'exploitation cherchera donc à minimiser le nombre d'accès à des pages situées sur le disque, de tels accès sont appelés défauts de page car la page fait défaut dans la mémoire.

Le système d'exploitation possède plusieurs moyens pour optimiser les performances, c'est à dire le taux d'utilisation du processeur ou son débit en nombre d'instructions par minutes. Son paramétrage permet de :
- fixer la taille des pages en nombre d'octets,
- fixer le nombre maximum de pages à gérer par le système (taille de la partition pour le "swapp" sur le disque),

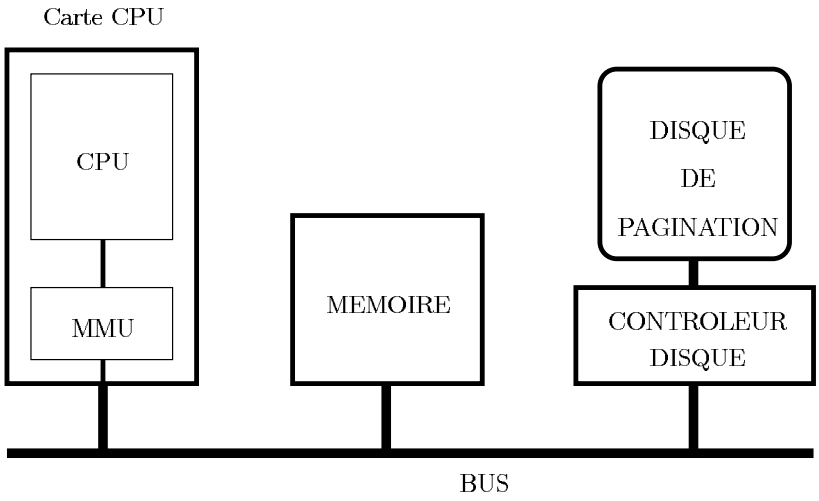

FIG. 3.1 – *Architecture d'une mémoire virtuelle paginée*

- choisir une stratégie de gestion des pages parmi un ensemble de stratégies préalablement programmées.

Pour notre étude, nous supposerons que la taille de la page est fixée, que le nombre total de pages est égal à N et que la capacité de la mémoire est de M pages. On supposera les pages distinctes (pas de duplication de pages) et identifiées par un numéro de page de 1 à N. Les pages seront sauvegardées dans N places de 1 à M pour les pages en mémoire et de $M+1$ à N pour les pages sauvegardées sur disque. En simplifiant, on pourrait caractériser l'état de la mémoire, à un instant donné, comme une permutation de $\{1, \cdots, N\}$.

	Mémoire Virtuelle							
	Mémoire			Disque				
Adresses	1	2	3	4	5	6	7	8
Pages	P_3	P_7	P_2	P_6	P_5	P_1	P_8	P_4

Par exemple, lorsque le système à une capacité de $N = 8$ pages et que sa mémoire peut contenir $M = 3$ pages, un état du système est déterminé par un N-uple, ici $E = (P_3, P_7, P_2, P_6, P_5, P_1, P_8, P_4)$. Cela signifie, dans cet exemple, que la page P_6 est à l'adresse 4.

FIG. 3.2 – *Un exemple de configuration d'un système*

Deux stratégies seront détaillées pour la gestion des pages

La stratégie Move-To-Front Dans cette stratégie, on suppose que lorsqu'une page est appelée, elle a toutes les chances d'être rappelée prochainement (phénomène de localisation des données d'un programme). Après le traitement effectué sur la page, on la replace en première position physique dans la mémoire. Dans ce cas il faut décaler les autres pages et, le cas échéant, sauvegarder sur le disque la page la plus ancienne en mémoire. Dans les systèmes d'exploitation, cette stratégie est appelée LRU (pour Last Recently Used).

	Mémoire Virtuelle								
	Mémoire			Disque					
Adresses	1	2	3	4	5	6	7	8	Etat
Pages	P_3	P_7	P_2	P_6	P_5	P_1	P_8	P_4	E
Pages	P_5	P_3	P_7	P_2	P_6	P_1	P_8	P_4	E_1

Après l'appel de la page P_5 le système passe dans un nouvel état E_1. Pour cette manipulation, il y a eu d'une part un défaut de page car la page P_2 était située sur le disque. Après traitement de la page P_5, celle-ci est replacée à l'adresse 1 et les autres pages sont décalées, la page P_2 est donc transferrée sur le disque.

FIG. 3.3 – *Application de la stratégie "Move-To-Front"*

Lorsque la page appelée est déjà en mémoire, il n'y a pas de défaut de page, seul l'ordre des pages en mémoire change.

Il faut remarquer que les opérations de décalage de page en mémoire ou sur le disque se font par permutation d'adresses physiques de pages (pointeurs), opérations qui s'effectuent en un temps négligeable devant le temps d'un transfert mémoire/disque.

La stratégie Move-Ahead Dans cette stratégie, on suppose que lorsqu'une page est appelée, on doit attendre un peu pour savoir si elle est fréquemment appelée ou pas. On essaye de ne garder en mémoire que les pages fréquemment appelées sur une période de temps. Pour cela chaque page est associée à un rang dans le système. Lorsque une page est appellée, elle avance d'un rang.

	Mémoire Virtuelle								
	Mémoire			Disque					
Adresses	1	2	3	4	5	6	7	8	Etat
Pages	P_3	P_7	P_2	P_6	P_5	P_1	P_8	P_4	E
Pages	P_3	P_7	P_2	P_5	P_6	P_1	P_8	P_4	E_2

Après l'appel de la page P_5 le système passe dans un nouvel état E_2. Pour cette manipulation, il y a eu d'une part un défaut de page car la page P_2 était située sur le disque. D'autre part, après traitement de la page P_5, celle-ci est replacée $Adresse(P_5) - 1 = 4$ et la page située à cette place est placée en position $Adresse(P_5) = 5$. Le contenu de la mémoire physique n'est donc pas modifié.

FIG. 3.4 – *Application de la stratégie "Move-Ahead"*

Pour synthétiser toute cette partie, on dira que le système est modélisé par un automate dont les états sont des permutations de $\{P_1, \cdots, P_N\}$. Les transitions de cet automate sont caractérisées par la stratégie de gestion de pages.

La difficulté de traitement d'un tel système est que la taille de l'espace d'état est très grande : $N!$. Par exemple dans un système simple tel que Linux sur un PC "classique", la taille de l'espace mémoire est de $16Mo$, la taille de l'espace disque réservé au Swapp est de $32Mo$ et la taille de

la page est de $4Ko$. Cela nous donne une taille d'espace d'état de 12000!, c'est à dire de l'ordre de 10^{44000}. Il faudra donc trouver des techniques pour réduire la complexité du problème.

3.2 Modélisation et réduction du problème

3.2.1 Modèle statistique du comportement dynamique du programme

On a vu que le comportement du programme pouvait être caractérisé par la suite des pages qu'il demande. Nous ferons les hypothèses probabilistes suivantes :

Indépendance des demandes de pages La séquence des pages demandées forme une séquence de variables aléatoires $\{M_n\}_{n\in\mathbb{N}}$, indépendantes à valeur dans $\{1,\cdots,N\}$. En l'absence de toute information supplémentaire sur le comportement du programme, nous considèrerons que la situation est "la plus difficile" à gérer, donc celle occasionnant un maximum de défauts de page.

Stationnarité des demandes La première information que l'on pourrait espérer sur la suite des appels de pages porte sur la fréquence empirique d'apparition d'une page. Nous supposerons donc que les demandes ont toutes la même distribution de probabilité sur $\{1,\cdots,N\}$ et sont indépendantes (voir le cours sur le principe du maximum d'entropie).

Ces hypothèses ne permettent pas de réduire la complexité du modèle. Il faut donc simplifier l'approche en ne considérant que la position d'une page donnée et de suivre son évolution au cours de l'exécution de l'algorithme.

Page la plus probable On supposera que l'une des pages, P_A, a la plus forte probabilité a d'appel et que, sans perte de généralité, toutes les autres pages seront demandées avec une même probabilité b, $b < a$.

Finalement, le comportement du programme sera modélisé par une suite de variables aléatoires $\{M_n\}_{n\in\mathbb{N}}$ indépendantes et de même loi de probabilité :

$$\mathbb{P}(M_n = P_A) = a,$$

$$\mathbb{P}(M_n = P_B) = b, \quad \text{pour} \quad B \neq A.$$

avec $a > b$ et $a + (N-1)b = 1$.

Le processus aléatoire étudié sera donc $\{X_n\}_{n\in\mathbb{N}}$ la position de la page A (la plus probable) au n-ième appel de page du programme. L'espace d'état a été considérablement réduit, car maintenant, l'automate sera de taille N au lieu de $N!$.

3.2.2 Stratégie "Move-To-Front"

D'après les hypothèses statistiques énoncées précédemment, $\{X_n\}$ est une chaîne de Markov à espace d'état $\{1,\cdots,N\}$ et de graphe de transition décrit en figure 3.5.

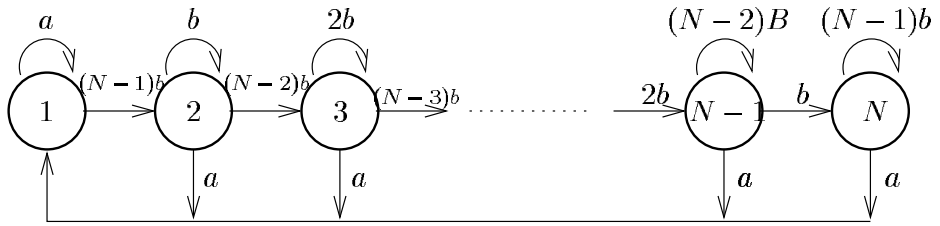

FIG. 3.5 – *Graphe de la chaîne de Markov associé à la stratégie "Move-To-Front"*

La matrice de transition associée est

$$P = \begin{bmatrix} a & (N-1)b & 0 & \cdots & \cdots & 0 \\ a & b & (N-2)b & \ddots & & \vdots \\ \vdots & 0 & 2b & (N-3)b & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & & \ddots & (N-2)b & b \\ a & 0 & \cdots & \cdots & 0 & (N-1)b \end{bmatrix}.$$

La résolution analytique de cette chaîne de Markov est donnée en Appendice A.

Exemple Numérique

Comme valeurs numériques, on prendra $N = 8$, $a = 0.3$ et $b = 0.1$. Le vecteur de probabilité stationnaire est alors (vecteur propre associé à la v.p. 1) :

$$\pi = [0.30\,,\, 0.23\,,\, 0.18\,,\, 0.12\,,\, 0.08\,,\, 0.05\,,\, 0.03\,,\, 0.01].$$

On rappelle que $\pi_i = \mathbb{P}(X_n = i)$ lorsque n est très grand.

3.2.3 Stratégie "Move-Ahead"

De même que dans la stratégie précédente $\{X_n\}$ est encore une chaîne de Markov dont le graphe de transition est décrit en figure 3.6

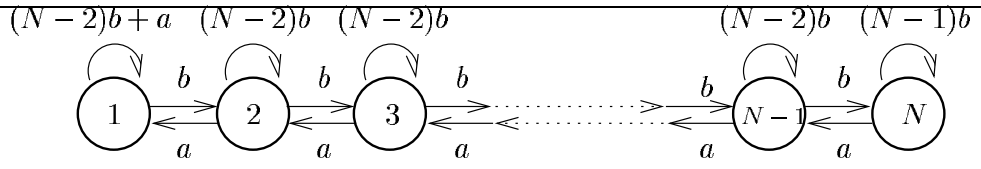

FIG. 3.6 – *Graphe de la chaîne de Markov associé à la stratégie "Move-Ahead"*

La matrice de transition associée s'écrit

$$P = \begin{bmatrix} a+(N-2)b & b & 0 & \cdots & \cdots & 0 \\ a & (N-2)b & b & \ddots & & \vdots \\ 0 & a & (N-2)b & b & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & (N-2)b & b \\ 0 & 0 & \cdots & 0 & a & (N-1)b \end{bmatrix}.$$

On trouvera les détails de la résolution analytique en Annexe B.

Exemple Numérique

En reprenant les mêmes valeurs numériques que précédemment on obtient :

$$\pi = [0.67\,,\, 0.22\,,\, 0.07\,,\, 0.02\,,\, 0.01\,,\, 0.01\,,\, 0.00\,,\, 0.00].$$

3.3 Interprétation des résultats

La première conclusion de notre étude (qui mériterait d'être prolongée et analysée plus en détail) est que les deux stratégies présentées sont *AUTO-ARRANGEANTES*. C'est à dire que la page P_A, appelée avec la plus grande fréquence, se trouvera, lorsque que l'on a atteint le régime stationnaire, le plus souvent dans la zone mémoire donc on réduira le nombre de défauts de page dûs à P_A. Cela se constate sur la distribution stationnaire des deux stratégies, car les π_i sont décroissants.

La deuxième conclusion porte sur l'aspect quantitatif. Dans les exemples numériques nous pouvons calculer la probabilité d'avoir un défaut de page lorsque la capacité de la mémoire est de M pages $p(M)$. Pour les deux stratégies, la probabilité de défaut de page $p(M)$ dû à la page A s'écrit :

$$p(M) = \sum_{i>M} \pi_i.$$

Ce qui donne les résultats numériques du tableau 3.1.

Taille Mémoire	Move to front	Move Ahead
0	1.00	1.00
1	0.70	0.33
2	0.47	0.11
3	0.29	0.04
4	0.17	0.02
5	0.09	0.01
6	0.04	0.00
7	0.01	0.00
8	0.00	0.00

TAB. 3.1 – *Probabilité de défaut de page dû à la page A en fonction de la taille mémoire*

A la lecture de ce tableau, on constate que la stratégie "Move-Ahead" semble bien meilleure que la stratégie "Move-To-Front", car la probabilité de défaut de page est bien plus faible dans "Move-Ahead" que dans "Move-To-Front".

Cependant, pour faire une comparaison plus globale des 2 stratégies il faudrait considérer tous les défauts de page possibles (la page A et les autres pages). Cette probabilité $p_g(M)$ s'écrit :

$$\begin{aligned} p_g(M) &= \sum_{i=1}^{N} \mathbb{P}(\text{ défaut de page } | P_A \text{ est en } i) \mathbb{P}(P_A \text{ est en } i) \\ &= \sum_{i=1}^{N} [(N-M)b + (a-b)\mathbb{1}_{i>M}]\pi_i. \end{aligned}$$

Ce qui se réécrit sous forme récurrente

$$\begin{aligned} p_g(0) &= 1, \\ p_g(M) &= p_g(M-1) - b - (a-b)\pi_M = 1 - Mb - (a-b)\sum_{i=1}^{M} \pi_i. \end{aligned}$$

Pour notre exemple numérique, on obtient le résultat numérique sur le tableau 3.2. On constate

Taille Mémoire	Move to front	Move Ahead
0	1.00	1.00
1	0.84	0.77
2	0.69	0.62
3	0.56	0.51
4	0.43	0.40
5	0.32	0.30
6	0.21	0.20
7	0.10	0.10
8	0.00	0.00

TAB. 3.2 – *Probabilité de défaut de page global en fonction de la taille mémoire*

alors que le résultat de l'analyse précédente est corrigé par le fait que la probabilité de défaut de page ne décroît que très faiblement en fonction de M. Dans ce cas, il faudrait remettre en question le modèle lui-même car il n'y a pas de raison d'avoir une loi uniforme sur les appels de pages différentes de la page 1 (c'est ce qui conduit à de fortes valeurs de probabilité de défaut de page). Il faudrait donc modifier le modèle et considérer des appels indépendants mais pour lesquels les fréquences d'appels de pages sont $f_1 > f_2 > \cdots > f_N$.

Dans le résultat que nous venons d'énoncer, il faut remarquer que l'interprétation dépend de l'hypothèse de stationnarité des appels de page. La vitesse de convergence vers l'état stationnaire est donc un point crucial pour l'évaluation de la stratégie. On sait, voir le cours sur les processus Markoviens, que la convergence vers la distribution stationnaire est géométrique de facteur le maximum des modules des valeurs propres de la matrice différentes de 1.

Dans notre exemple, pour la stratégie "Move-To-Front", le spectre de la matrice est

$$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1\}$$

et donc la convergence se fait en 0.6^n, ce qui est assez rapide. Pour la stratégie "Move-Ahead" le spectre de la matrice est

$$\{0.28, 0.36, 0.47, 0.6, 0.73, 0.84, 0.92, 1\}$$

et la vitesse de convergence vers l'état stationnaire est en 0.92^n ce qui est plus lent que la convergence pour la stratégie "Move-To-Front" !

3.4 Début de conclusion

L'analyse des performances des deux modèles a montré que les deux stratégies étaient auto-arrangeantes et par conséquent s'adaperont bien au fur et à mesure de l'exécution d'un programme.

Cependant, lorsque les appels sont "statistiquement stabilisés" (les fréquences d'appels aux pages restent à peu près constantes) la stratégie "Move-Ahead" est meilleure que la stratégie "Move-To-Front". Cette stratégie positionnera en mémoire les pages demandées moins rapidement que pour la stratégie "Move-To-Front", il y aura donc beaucoup de défauts de page en début d'exécution du programme.

Par contre, si l'on observe des variations des fréquences empiriques d'appels de page de l'ordre de la vitesse de convergence vers l'état stationnaire, la stratégie "Move-Ahead" sera inefficace (le temps de positionnement d'une page en mémoire sera trop long par rapport à la durée d'utilisation de la page par le programme) et la stratégie "Move-To-Front" sera efficace.

A vous de trouver, en situation réelle, le bon compromis !

Appendice A : Formules analytiques pour la politique "Move-To-Front"

L'automate de la politique Move-To-Front décrit en figure 3.5 montre que la chaîne de Markov associée est irréductible (graphe d'état fortement connexe) et apériodique (pas de décomposition périodique à cause des boucles). Le théorème sur la convergence des chaînes de Markov[1] 10

peut donc être appliqué : il existe une unique distribution stationnaire π vérifiant $\pi P = \pi$,

$$\lim_{n\to+\infty} \mathbb{P}(X_n = i) = \pi_i \quad \text{et} \quad \lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_n=i} = \pi_i \quad \text{P-p.s.}$$

Il faut donc calculer analytiquement la valeur de π. Le graphe de l'automate nous montre que les π_i doivent vérifier[2]

$$(N-1-i)b\pi_{i-1} = (a+(N-i)b)\pi_i, \quad \text{pour} \quad i > 1.$$

On en déduit que

$$\pi_i = \frac{(N-1-i)b}{a+(N-i)b}\pi_{i-1} = \frac{(N-1-i)\cdots(N-2)(N-1)b^{i-1}}{(a+(N-i)b)\cdots(a+(N-2)b)(a+(N-1)b)}\pi_1.$$

1. Pour une introduction élémentaire aux chaînes de Markov on pourra consulter [8] p 46 ou [32], pour des approfondissements et des applications à l'analyse d'algorithmes [20].

2. La probabilité d'entrer dans un état en régime stationnaire est égale à la probabilité d'en sortir.

π_1 étant obtenu par l'équation de normalisation,

$$1 = \sum_{i=1}^{N} \pi_i = \pi_1 \sum_{i=1}^{N} \frac{(N-1-i)\cdots(N-2)(N-1)b^{i-1}}{(a+(N-i)b)\cdots(a+(N-2)b)(a+(N-1)b)}.$$

D'où

$$\pi_1 = \left[\sum_{i=1}^{N} \frac{(N-1-i)\cdots(N-2)(N-1)b^{i-1}}{(a+(N-i)b)\cdots(a+(N-2)b)(a+(N-1)b)}\right]^{-1}.$$

De plus, pour cette matrice de transition, il est facile de vérifier que le polynôme caractéristique s'écrit

$$Q(X) = det(P - X\,Id) = -X(b-X)(2b-X)\cdots((N-2)b-X)(1-X).$$

On en déduit immédiatement la seconde valeur propre $(N-1)b$ qui fixe la vitesse de convergence vers le régime stationnaire.

Appendice B : Formules analytiques pour la politique "Move-Ahead"

Comme dans le cas de la politique Move-To-Front, l'automate de la politique Move-Ahead décrit en figure 3.6 montre que la chaîne de Markov associée est irréductible et apériodique. Le théorème sur la convergence des chaînes de Markov peut donc également être appliqué. Il faut donc calculer analytiquement la valeur de π. Le graphe de l'automate nous montre que les π_i doivent vérifier

$$b\pi_{i-1} = a\pi_i, \quad \text{pour} \quad i > 1.$$

On en déduit que

$$\pi_i = \frac{b}{a}\pi_{i-1} = (\frac{b}{a})^{i-1}\pi_1,$$

π_1 étant obtenu par l'équation de normalisation,

$$1 = \sum_{i=1}^{N} \pi_i = \pi_1 \sum_{i=1}^{N} (\frac{b}{a})^{i-1}.$$

D'où

$$\pi_1 = \frac{1-\frac{b}{a}}{1-(\frac{b}{a})^N}.$$

Par suite,

$$\pi_i = (\frac{b}{a})^{i-1}\frac{1-\frac{b}{a}}{1-(\frac{b}{a})^N}.$$

A l'état stationnaire la position de la page P_A, page la plus probable, suit une loi géométrique de paramètre $\frac{b}{a}$.

Le calcul du polynôme caractéristique reste cependant difficile analytiquement et n'a été effectué que sur les exemples numériques en utilisant un logiciel adapté.