Université de Strasbourg
UFR Mathématique et Informatique

M1 Informatique
2021/2022

Algorithmes Distribués

TD n° 3 : Interblocages

Exercice 1 : Caractérisation d'une situation d'interblocage et introduction à sa prévention

On dispose d'un ensemble de processus, et de ressources critiques (1 unité par ressource) nécessaires à l'exécution de ces processus : bloc disque, imprimante, modem, etc. Chaque processus, lors de son exécution, demande l'allocation de ressources. Les ressources sont libérées en fin d'exécution du processus (voir table 1).

P_1	P_2	P_3
demander(bloc disque)	demander(modem)	demander(imprimante)
demander(modem)	demander(imprimante)	demander(bloc disque)
exécution	**exécution**	**exécution**
libérer(modem)	libérer(imprimante)	libérer(bloc disque)
libérer(bloc disque)	libérer(modem)	libérer(imprimante)

TABLE 1 – Exécution de 3 processus utilisant 3 ressources critiques.

Q 1. Montrez qu'il existe un risque d'interblocage avec un tel scénario. Rappelez les 3 conditions nécessaires à l'apparition d'un interblocage.
Q 2. Si l'interblocage se produisait, comment pourrait-on le résoudre ?
Q 3. Proposez une solution de prévention statique de l'interblocage pour ce cas.

Exercice 2 : Prévention d'interblocage (modèle basé sur les ressources) – *Wait or Die vs. Wound or Wait*

Application de l'algorithme de Rosenkrantz, Stearns et Lewis au scénario détaillé ci-dessous.
Soit quatre transactions avec les séquences d'exécution suivantes :

Transaction	Débute au temps	Séquence à exécuter
T1	1	L(A), R(A), W(A), L(B), W(B), U(A), U(B)
T2	2	L(A), L(C), R(C), W(A), U(C), U(A)
T3	3	L(B), R(B), L(C), W(C), U(B), U(C)
T4	4	L(C), R(C), L(A), W(A), U(C), U(A)

Avec les temps d'exécution de chaque opération (en unité processeur) :

Opération	Temps processeur	Définition
L(X)	1	Demande de verrouillage sur la variable X
R(X)	3	Lecture de la variable X
W(X)	3	Écriture sur la variable X
U(X)	1	Libération de la variable X
Restart()	3	Redémarrage
KilledBy(P)	3	Interruption par le processus P
Wait(X, P)	_	Attente de la variable X détenue par le processus P

Q 1. Représentez le graphe des dépendances de cette exécution. Existe-t-il un interblocage ?
Q 2. Déroulez la séquence d'exécution décrite précédemment en utilisant la méthode wait or die, puis la méthode wound or wait. Expliquez comment chaque méthode a évité l'interblocage.
Q 3. Comparez les deux exécutions et dites quels sont les avantages et les inconvénients de l'une par rapport à l'autre en termes de nombre d'interruptions et de temps processeur.
Q 4. (optionnel) Que se passe-t-il si on change la séquence du processus 4 par : L(A), L(D), R(D), W(A), U(D), U(A) ? Qu'est-ce que cela indique sur le comportement des deux méthodes en terme de temps d'exécution ?

Rappel : fonctionnement de base des algorithmes Wait or Die et Wound or Wait
Soit $TS(T_i)$ l'estampille à laquelle la transaction a démarré.
Soit deux transactions T_i et T_j, si T_i essaye d'accéder à la ressource R, mais que celle ci est déjà verrouillée par le processus T_j, la règle à appliquer est la suivante, en fonction de la méthode choisie :

- **wait-die :** si $TS(T_i) < TS(T_j)$ (T_i est plus ancien que T_j) alors T_i peut attendre, sinon T_i est interrompue (tuée) et redémarre plus tard avec la même estampille de démarrage.
- **wound-wait :** si $TS(T_i) < TS(T_j)$ (T_i est plus ancien que T_j) alors T_j est interrompue (T_i blesse T_j) et est redémarrée plus tard avec la même estampille de démarrage, sinon T_i peut attendre.

Exercice 3 : Algorithme de détection d'interblocage (modèle basé sur la communication de messages)

Application de l'algorithme de Chandy, Misra et Haas (dont le code est fourni en annexe 1) au scénario dont l'état initial est le suivant :

- le processus P1 est passif, et attend un message en provenance de P2 ou de P3 ;
- le processus P2 est passif, et attend un message en provenance de P4 ;
- le processus P3 est passif, et attend un message en provenance de P1 ou de P4 ;
- le processus P4 s'exécute.

Q 1. Tracez le graphe des dépendances à l'état initial. Est-ce qu'il inclut une CFTC (composante fortement connexe terminale) ?
Q 2. Complétez les messages de requête et de réponse pour les calculs diffusants, ainsi que l'évolution des variables locales des processus selon l'algorithme de Chandy, Misra et Haas, dans la séquence d'événements suivante (tracez aussi l'évolution du graphe des dépendances au fur et à mesure du déroulement du scénario) :

1. P1 initie son premier calcul diffusant :
   dernier1(…) ← … ; attendre1(…) ←… ; num1(…) ← … ;
   P1 envoie requête (…, …, …, …) et requête (…, …, …, …)
2. P2 reçoit requête (…, …, …, …) :
   père2(…) ← … ; dernier2(…) ← … ; attendre2(…) ← … ; num2(…) ← … ;
   P2 envoie requête (…, …, …, …)
3. P3 reçoit requête (…, …, …, …) :
   père3(…) ← … ; dernier3(…) ← … ; attendre3(…) ← … ; num3(…) ← … ;
   P3 envoie requête (…, …, …, …) et requête (…, …, …, …)
4. P4 envoie un message à P3
5. P1 reçoit requête (…, …, …, …) : P1 envoie réponse (…, …, …, …)
6. P4 passe de l'état actif à l'état passif et attend un message de P2
7. P4 reçoit requête (…, …, …, …) envoyée au (2) :
   père4(…) ← … ; dernier4(…) ← … ; attendre4(…) ← … ; num4(…) ← … ;
   P4 envoie requête (…, …, …, …)
8. P4 reçoit requête (…, …, …, …) envoyée au (3) :
   P4 envoie réponse (…, …, …, …)
9. P3 reçoit le message de P4 envoyé au (4) :
   …… attendre3(…) ← … ;
10. P4 initie son premier calcul diffusant :
    dernier4(…) ← … ; attendre4(…) ← … ; num4(…) ← … ;
    P4 envoie requête (…, …, …, …)
11. P2 reçoit requête (…, …, …, …) envoyée au (7) :
    P2 envoie réponse (…, …, …, …)
12. P3 reçoit réponse (…, …, …, …) envoyée au (8) :
    …..
13. P2 reçoit requête (…, …, …, …) envoyée au (10) :
    père2(…) ← … ; dernier2(…) ← … ; attendre2(…) ← … ; num2(…) ← … ;
    P2 envoie requête (…, …, …, …)
14. P4 reçoit réponse (…, …, …, …) envoyée au (11) :
    num4(…) ← …
    P4 envoie réponse (…, …, …, …)

15. P4 reçoit requête (. . . , . . . , . . . , . . .) envoyée au (13) :
    P4 envoie réponse (. . . , . . . , . . . , . . .)
16. P2 reçoit réponse (. . . , . . . , . . . , . . .) envoyée au (14) :
    num2(. . .) ← . . .
    P2 envoie réponse (. . . , . . . , . . . , . . .)
17. P1 reçoit réponse (. . . , . . . , . . . , . . .) envoyée au (16) :
    num1(. . .) ← . . .
18. P2 reçoit réponse (. . . , . . . , . . . , . . .) envoyée au (15) :
    num2(. . .) ← . . .
    P2 envoie réponse (. . . , . . . , . . . , . . .)
19. P4 reçoit réponse (. . . , . . . , . . . , . . .) :
    num4(. . .) ← . . .
    . . . .

Q 3. Montrez que si l'initiateur d'un calcul diffusant se déclare interbloqué, alors il appartient à un ensemble de processus interbloqués.

<u>Exercice 4</u> : Introduction à l'évitement d'interblocage – définitions et sûreté

On considère un système comprenant 15 cases de mémoire (ressources banalisées) partagées par 3 processus.

À l'instant $t1$, on a l'état suivant avec $Disponible = (0)$:

Processus	Détenu	Besoin
P_1	3	0
P_2	6	2
P_3	6	2

À l'instant t2, on a l'état suivant avec $Disponible = (0)$:

Processus	Détenu	Besoin
P_1	5	1
P_2	5	2
P_3	5	3

À l'instant t3, on a l'état suivant avec $Disponible = (3)$:

Processus	Détenu	Besoin
P_1	2	1
P_2	7	2
P_3	3	3

Q 1. Formalisez les termes suivants :
— Quand est-ce qu'un processus P_i est dit *bloqué* à l'état l ?
— Quand est ce que le système est dit en *interblocage* à l'état l ?

Q 2. Pour les instants $t1$, $t2$ et $t3$, dites s'il y a interblocage. On montrera que l'état du système est sûr ou non.

On suppose maintenant que la demande maximale de chacun des processus est de 10 cases mémoire et le besoin réel est inconnu.

Q 3. Les états du système aux instants $t1$, $t2$ et $t3$ sont-ils sûrs ?

Exercice 5 : Évitement d'interblocage avec l'algorithme du banquier – facultatif (non évalué en CC)

On considère 3 ressources R_1, R_2 et R_3, disposant chacune de 3 unités, et 3 processus devant exécuter les tâches suivantes :

Processus	Tâche	Demande	Rend
P_1	$T_{1,1}$	(1,0,2)	(0,0,0)
	$T_{1,2}$	(0,2,0)	(1,2,2)
P_2	$T_{2,1}$	(1,2,0)	(1,1,0)
	$T_{2,2}$	(1,0,0)	(1,0,0)
	$T_{2,3}$	(0,0,2)	(0,1,2)
P_3	$T_{3,1}$	(1,1,0)	(1,0,0)
	$T_{3,2}$	(1,1,0)	(0,0,0)
	$T_{3,3}$	(1,0,0)	(1,0,0)
	$T_{3,4}$	(0,0,2)	(1,2,2)

Q 1. Montrez que la séquence d'exécution $T_{1,1}\ T_{2,1}\ T_{3,1}\ f_{T_{3,1}}\ f_{T_{2,1}}\ T_{2,2}\ T_{3,2}\ f_{T_{3,2}}\ f_{T_{2,2}}\ T_{3,3}\ f_{T_{3,3}}\ f_{T_{1,1}}$ aboutit à un interblocage.
Q 2. En appliquant l'algorithme du banquier au cours de cette exécution, quel serait le dernier état sûr ?
Q 3. Utilisez l'algorithme du banquier pour proposer une exécution réalisable.

On connait à présent les temps d'exécution de chaque tâche (cf. table 2).

Processus	Tâche	Durée
P_1	$T_{1,1}$	10 sec.
	$T_{1,2}$	2 sec.
P_2	$T_{2,1}$	2 sec.
	$T_{2,2}$	3 sec.
	$T_{2,3}$	1 sec.
P_3	$T_{3,1}$	1 sec.
	$T_{3,2}$	2 sec.
	$T_{3,3}$	1 sec.
	$T_{3,4}$	1 sec.

TABLE 2 – Temps d'exécution des tâches.

Q 4. Quel temps prendrait l'exécution calculée à l'aide de l'algorithme du banquier ?
Q 5. Est-il possible de faire mieux ? Pourquoi ?

Fonctionnement de l'algorithme du banquier
Pour passer d'un état sûr s_l à un état s_{l+1}, on vérifie d'abord que s_{l+1} est un état sûr.
- Si la transition $s_l \rightarrow s_{l+1}$ est une libération de ressources (i.e. fin d'une tâche), alors on peut autoriser la transition ;
- Si la transition $s_l \rightarrow s_{l+1}$ est une demande d'allocation de ressources, alors :
  - on calcule l'état s_{l+1} en construisant le pire état virtuel associé $t_l = [Besoin^t(l), Detenu^t(l)]$, avec
    - $Besoin_i^t(l) = MAX_i - Detenu_i^t(l)$ si $Besoin_i^t(l) + Detenu_i^t(l) > 0$
    - $Besoin_i^t(l) = 0$ sinon.
  - Si l'état t_l ne contient pas un interblocage alors la transition est validée.
  - Sinon, la demande est rejetée (on ne peut pas conclure).

Annexe 1. Algorithme de prévention de l'interblocage de Chandy, Misra et Haas

Variables du processus P_i : P_i mémorise 4 tableaux dont les indices correspondent à tous les processus du système :

— dernier : tableau [1..n] de entier initialisé à 0 : contient les numéros d'ordre des derniers calculs diffusants lancés par les processus du système tels que P_i les connaît. Ainsi dernier[j] contient la plus grande valeur m reçue par P_i dans un message requête (j, m, l, i).

— num : tableau [1..n] de entier : nombre (j) indique combien de messages de la forme requête (j, dernier(j), i, k) émis par P_i n'ont pas encore reçu un message de type réponse (j, dernier(j), k, i) en provenance des processus Pk de EDi.

— attendre : tableau [1..n] de booléen initialisé à faux : attendre(j) est un booléen qui vaudra vrai si P_i est resté tout le temps inactif depuis le passage du premier message de type requête (j, dernier(j), *, i) (l'inactivité considérée est celle du processus vis-à-vis de son calcul propre et non du point de vue de la détection de l'interblocage).

— père : tableau [1..n] de entier : père(j) contient le numéro du processus ayant expédié vers P_i le premier message de la détection (j, dernier(j)). (père(j) est le numéro du processus qui est le père de P_i dans le calcul diffusant relatif à cette détection).

On associe aussi à P_i une variable qui définit à tout instant son ensemble de dépendance (cette variable est mise à jour par le processus P_i lorsqu'il se met en attente de message), et une variable état qui est mise à jour quand le processus devient actif ou passif :

— ens-dep$_i$: ensemble de dépendance 1..n ;

— état$_i$: actif, passif ;

Algorithme

Lorsque état$_i$ = PASSIF et décision de P_i d'initier un calcul diffusant :

début
  dernier(i) ← dernier(i) +1;
  attendre(i) ← vrai;
  pour j ∈ ens-dep$_i$ **faire**
    envoyer requête (i, dernier(i), i, j) à P_j;
  fin
  num(i) ← card (ens-dep$_i$);
fin

Lorsque P_k reçoit un message différent de requête/réponse (devient actif) :

début
  état$_k$ ← actif;
  ∀ i, attendre(i) ← faux;
  ignorer toutes les requêtes et réponses reçues tant que le processus s'exécute;
fin

Lorsqu'un processus P_k reçoit une requête (i, m, j, k) et état$_k$ = passif :

```
début
    si m > dernier(i) alors
        dernier(i) ← m;
        père(i) ← j;
        attendre(i) ← vrai;
        pour r ∈ ens-dep_k faire
            envoyer requête (i, m, k, r) à P_r;
        fin
        num(i) ← card (ens-dep_k);
    sinon
        si attendre(i) et m = dernier(i) alors
            envoyer réponse (i, m, k, j) à P_j;
        fin
    fin
fin
```

Lorsqu'un processus P_k reçoit réponse (i, m, r, k) et état$_k$ = passif :

```
début
    si m = dernier(i) et attendre(i) alors
        num (i) ← num (i) – 1 ;
        si num (i) = 0 alors
            si i = k alors
                déclarer P_k interbloqué
            sinon
                j ← père (i) ;
                envoyer réponse (i, m, k, j) à P_j ;
            fin
        fin
    fin
fin
```