Interblocage

Conditions d'apparition

1. Chaque ressource est utilisable exclusivement par une tâche et une seule, son partage donnant des résultats incohérents
2. Chaque tâche ne peut progresser que si elle obtient les ressources requises dynamiquement par des requêtes successives
3. Les ressources déjà allouées à une tâche restent nécessaire à la tâche qui les a reçus et donc ne peuvent être réquisitionnées.
4. Des tâches sont en attente de ressources allouées à d'autres tâches et il s'est formé un cycle des tâches en attente.

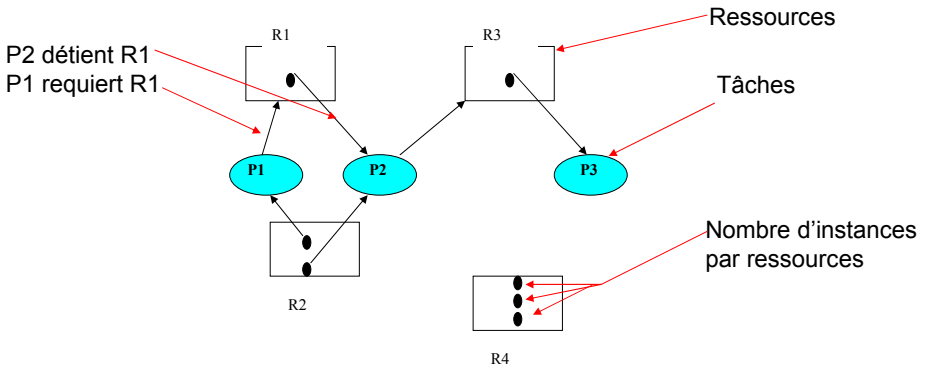
Approches de résolution du problème

- Détection puis guérison par sacrifice d'une tâche
  - Les demandes et les allocations sont enregistrées et un cycle est recherché dans un graphe à n nœuds
  - Si un cycle est détecté, il est rompu en tuant une tâche et récupérant ses ressources.
- Prévention statique
  - Toute tâche fait en une seule fois la demande de toutes les ressources nécessaires.
  - Les ressources peuvent être regroupées en classe et ordonnées selon l'importance de chaque classe. Elles sont alors allouées globalement, mais par classe, selon l'ordre fixé : cet ordre fixe d'allocation empêche la formation de cycle.

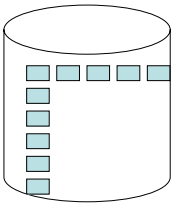
Approches (suite)

- Prévention dynamique par évitement : attente et retard de service
  - Eviter la destruction de tâche
  - Estimer le comportement des tâches par l'annonce du maximum de ressources que chacune peut demander
- Rôle de l'allocateur
  - Vérifier, avant d'honorer une nouvelle requête, si le système conserve la possibilité de répondre à toutes les demandes futures.
  - Dans le cas contraire, il y a risque d'interblocage et l'allocateur retarde la requête en bloquant la tâche
  - L'allocateur pilote le système en l'obligeant à rester dans des états, appelés sains, qui évitent d'évoluer vers un interblocage

Représentation



Exemples



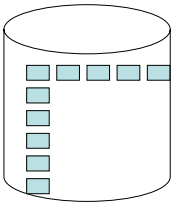
50 clusters disques

Disponible = 50 – (14+7+9+14)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	**3**	**2**
3	9	**4**	**1**
6	5	7	**2**
7	5	2	**2**
1		2	9
		1	
TOTAUX 22	26	21	23

t1	**Allocation**	**Max (cumul)**	**Disponible**
P1	14	21	6
P2	7	16	
P3	9	16	
P4	14	23	

Aucun besoin ne peut être satisfait avec le disponible restant



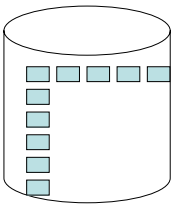
50 clusters disques

Disponible = 50 – (14+21+2+12)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	3	**2**
3	**9**	4	**1**
6	**5**	7	**2**
7	5	2	2
1		2	9
		1	
TOTAUX 22	26	21	23

t2	**Allocation**	**Max (cumul)**	**Disponible**
P1	14	21	1
P2	21	26	
P3	2	5	
P4	12	14	

Aucun besoin ne peut être satisfait avec le disponible restant



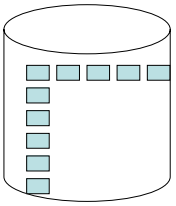
50 clusters disques

Disponible = 50 – (8+16+9+14)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	**3**	**2**
3	**9**	**4**	**1**
6	5	7	**2**
7	5	2	**2**
1		2	9
		1	
TOTAUX 22	26	21	23

t3	**Allocation**	**Max (cumul)**	**Disponible**
P1	8	14	3
P2	16	21	
P3	9	16	
P4	14	23	

Aucun besoin ne peut être satisfait avec le disponible restant



50 clusters disques

Disponible = 50 – (14+7+2+9)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	3	**2**
3	9	4	1
6	5	7	2
7	5	2	2
1		2	9
		1	
TOTAUX 22	26	21	23

MAX = 26

t4	**Allocation**	**Max (cumul)**	**Disponible**
P1	14	14	18
P2	7	16	
P3	2	9	
P4	9	14	

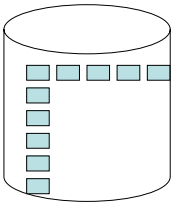
P2, P3 ou P4 peuvent être servies

- Servir Pi tel que le disponible restant permette d'atteindre le maximum pour une des tâches restantes

  Servir P2 (besoin = 9) => (disponible = 9) => aucune tâche ne peut atteindre la valeur max = 26 par la suite

  Servir P3 (besoin = 7) => disponible = 11 => idem

  Servir P4 (besoin = 5) => disponible = 13 => P1 peut atteindre 26 (14 +12) et donc P4 peut être servie.



50 clusters disques

Disponible = 50 – (14+7+2+14)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	3	**2**
3	9	4	**1**
6	5	7	**2**
7	5	2	**2**
1		2	9
		1	
TOTAUX 22	26	21	23

MAX = 26

t5	**Allocation**	**Max (cumul)**	**Dispo nible**
P1	14	21	13
P2	7	16	
P3	2	9	
P4	14	23	

Généralisation à n classes de ressources

- Soit (P0, P1, P2, P3, P4) 5 tâches et (A, B, C = 10, 5, 7) 3 classes de ressources.

t0	**Allocation**	**Max**	**Dispo**
	A B C	**A B C**	**A B C**
P0	0 1 0	7 5 3	3 3 2
P1	2 0 0	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

Besoin = Max -Alloc

	A B C
P0	7 4 3
P1	1 2 2
P2	6 0 0
P3	0 1 1
P4	4 3 1

1. Work = dispo [3,3,2]
2. Chercher i tel que Besoin <= Work (i = 1)
3. Work = Work + Alloc(1)
   Work = [5, 3, 2] et P1 pris
4. Goto 2
5. Chercher i tel que Besoin <= Work (i = 3)
6. Work = Work + Alloc(3)
7. Work = [7, 4, 3] et P3 pris
8. Goto 2
9. Chercher i tel que Besoin <= Work (i = 0)
10. Work = Work + Alloc(0)
    Work = [7, 5, 3] et P0 pris
11. Goto 2
12. Chercher i tel que Besoin <= Work (i = 2)
13. Work = Work + Alloc(2)
    Work = [10, 5, 5] et P2 pris
14. Goto 2

15. Chercher i tel que Besoin <= Work (i = 4)
16. Work = Work + Alloc(4)
    Work = [10, 5, 7] et P4 pris
17. Fin car tous les i ont été trouvés

La séquence <P1, P3, P0, P2, P4> a permis de satisfaire toutes les demandes ; le système est dit dans un état sain.

Soit à t1 la requête P1 = (1, 0, 2)

t1	Allocation	Max	Dispo
	A B C	A B C	A B C
P0	0 1 0	7 5 3	2 3 0
P1	3 0 2	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

	Besoin = Max -Alloc
	A B C
P0	7 4 3
P1	0 2 0
P2	6 0 0
P3	0 1 1
P4	4 3 1

- La séquence (P1, P3, P4, P0, P2) laisse le système dans un état sain
- Work = [2 3 0], [5 3 2], [7 4 3], [7 4 5], [7 5 5], [10 5 7]

Soit à t2 la requête P4 = (3, 3, 0) impossible

t2	Allocation	Max	Dispo
	A B C	A B C	A B C
P0	0 1 0	7 5 3	2 3 0
P1	3 0 2	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

	Besoin = Max -Alloc
	A B C
P0	7 4 3
P1	0 2 0
P2	6 0 0
P3	0 1 1
P4	4 3 1

Soit à t2 la requête P0 = (0, 2, 0) possible mais

t2	Allocation	Max	Dispo

	A B C	A B C	A B C
P0	0 3 0	7 5 3	2 1 0
P1	3 0 2	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

Besoin = Max -Alloc

	A B C
P0	7 2 3
P1	0 2 0
P2	6 0 0
P3	0 1 1
P4	4 3 1

- Work = [2 1 0], aucun i ne peut satisfaire $besoin_i$ <= Work donc satisfaire cette requête conduirait à un état dit non sain