DS 2011 Systèmes avancés

Damien Crémilleux

4 juin 2014

1 Questions de cours

1.1 Deadlocks

Donnez une définition courte d'un interblocage et un exemple autre que celui du court Un interblocage (deadlock en anglais) est un phénomène qui peut survenir en programmation concurrente. L'interblocage se produit lorsque deux processus concurrents s'attendent mutuellement. Les processus bloqués dans cet état le sont définitivement.

Un exemple concret d'interblocage peut se produire lorsque deux processus légers essayent d'acquérir deux verrous dans un ordre différent. Ainsi, si par exemple il y a deux mutex (nommés M1 et M2) et les deux processus légers suivants :

```
TacheA :
  Obtenir M1
  Obtenir M2
  Action necessitant les deux verrous
  Rendre M2
  Rendre M1

TacheB :
  Obtenir M2
  Obtenir M1
  Action necessitant les deux verrous
  Rendre M1
  Rendre M2
```

Un interblocage est possible. En effet, si par exemple, la séquence d'opération suivante survient :

- La TâcheA obtient M1.
- La TâcheB obtient M2.
- La TâcheA attend pour obtenir M2 (qui est entre les mains de TâcheB).
- La TâcheB attend pour obtenir M1 (qui est entre les mains de TâcheA).

Dans cette situation, les deux tâches (TâcheA et TâcheB) sont définitivement bloquée

Quelle est la différence avec un blocage de type *livelock* ? Un *livelock* est un interblocage durant lequel les états des processus impliqués est en changement constant, mais il n'y a pas de progression dans l'exécution.

Comment détecte t'on un interblocage ? Il est possible de détecter des interblocages de plusieurs manières :

- Utilisation d'un timer (chien de garde). Un timer est régulièrement déclenché, et permet de détecter des interblocages probables.
- Parcours du graphe des attentes. Ce graphe, qui représente les ressources attendues par les différents processus ne doit pas contenir de cycle. Si c'est le cas, alors il y a un interblocage. Il est donc nécessaire de parcourir ce graphe à intervalle régulier.
- Il est également possible de les détecter à l'aide d'un algorithme reposant sur l'utilisation de matrices.

Quels problèmes posent la résolution d'un interblocage en particulier dans un système critique ? Afin de résoudre un problème d'interblocage, il existe plusieurs solutions : le reboot automatique, tuer un processus afin de libérer les autres, ou réquisitionner une ressource. Ces solutions ne sont pas compatibles avec les systèmes critiques dont le bon fonctionnement est essentiel. Elles sont seulement compatibles avec les systèmes d'usage général.

Quelle est l'idée des ressources ordonnées ? Les ressources ordonnées sont un moyen de prévention des interblocages. Dans ce but, les différentes ressources sont numérotées, et les processus vont demander les ressources dans l'ordre. Ce mécanisme empêche l'apparition de cycle et donc d'interblocage. Les ressources sont classées de façon inversement proportionnelle à leur coût (les ressources les plus prioritaires sont donc demandées en premier).

Pourquoi cette solution d'évitement d'interblocage n'est pas toujours utilisée ? Cette solution n'est pas toujours utilisée car il est souvent difficile de trouver un ordre satisfaisant devant le nombre de ressource et leur diversité.

1.2 Mémoire virtuellement partagée

Comment est utilisée la mémoire physique dans un système à mémoire virtuellement partagée ? Dans un système à mémoire virtuellement partagée, la mémoire physique peut être utilisé de façon partagée, ou de façon distribuée. En mode partagé (exemple : *Uniform Memory Access*), tous les processeur partagent la mémoire physique de façon uniforme. Les temps d'accès sont donc indépendants du processeur faisant la requête ou de l'emplacement réelle de la mémoire. En mode distribué (exemple : *Non-Uniform Access Memory*), les processeurs peuvent directement adresser les différentes mémoires.

Quel est le rôle d'une mémoire virtuellement partagée ? En programmation parallèle, il est nécessaire de partager des données entre plusieurs processus. Le partage de mémoire à l'aide de la mémoire virtuellement partagée un moyen de rendre accessible les différentes données. Il s'agit donc d'un mécanisme pour abstraire une mémoire globale.

Pourquoi le protocole par invalidation est-il plus intéressant que le protocole de mise à jour ? Dans le cas du protocole par invalidation, en cas d'écriture, les copies sont simplement invalidées. Le changement est ensuite

simplement effectué en local. Dans le cas du protocole de mise à jour, toutes les copies sont mises à jour et non pas simplement invalidées. Le problème de ce dernier protocole est le très grand nombre de mises à jour envoyées à chaque changement. C'est cette raison qui rend le protocole par invalidation plus intéressant que le protocole de mise à jour.

1.3 Organisations virtuelles (VO)

Dans quel cadre a-t'on besoin de définir une organisation virtuelle ?

Quel est son rôle ?

Pourquoi l'utilisation des droits Unix n'est pas satisfaisante dans un système distribué à large échelle (Grid ou Cloud) ? Dans un système distribué à large échelle, comme les Grids ou les Clouds, il n'est pas satisfaisant d'utiliser les droits Unix. En effet, les nombres d'acteurs (de clients et de serveurs), empêche une utilisation correcte de ce moyen.

1.4 Systèmes pair à pair

Expliquez brièvement les différences entre Pair à Pair et serveur centralisé : avantages/inconvénients, fiabilité, résistance à la charge, nombres de messages échangés, ...

Y a-t'il des avantages non techniques au Pair à Pair ?

2 Question rédactionnelle longue : Mémoire virtuellement partagée

Le schéma expliquant les différents échanges de messages est situé page 21 du poly *Memory Management*.

On suppose qu'un processus est en charge du rôle de manager. Les requêtes d'accès aux pages sont donc envoyées à ce processus. La structure de données est la suivante :

- Deux tableau de p (nombre de pages) entrées : ptable[] et info[] ;
- Pour chaque entrée p du tableau info[] : le propriétaire de la page (info[p].owner), l'ensemble des processus qui ont une copie de la page (info[p].copyset), un verrou pour la synchronisation (info[p].lock) ;
- Pour chaque entrée p du tableau ptable[] : un verrou pour la synchronisation (ptable[p].lock), le type d'accès à la page (ptable[p].access).

Pour la suite de la question, on supposera que le processus P1 demande une page p. Cette page est détenue par le processus P4. P3 possède une copie de cette page et P2 est le manager. Le code exécuté par P1 est donc le code de gauche. Les autres processus exécutent le code de droite.

Le déroulement est le suivant :

- P1 demande la page p. Il commence donc par verrouiller cette page (*lock(ptable[p].lock)* et envoie ensuite donc une requêtes au manager (en vérifiant que ce n'est pas lui même).

– Le manager va commencer par invalider les copies de la page demandée (*invalidate(p,info[p].copyset)*). La copie possédée par P3 est donc invalidée. L'ensemble des processeur qui possède une copie de la page est mis à vide (*info[p].copyset* = {}).
– Le manager va ensuite envoyer une requête au propriétaire actuel de la page, c'est-à-dire P4 avec *send request to info[p].owner*. Le propriétaire envoie la page p au processeur P1 et invalide sa propre copie.
– Une fois la page reçu, P1 envoie au manager un acquittement, avant de libérer le verrou sur ptable[p] (*ptable[p].lock*).