CHAPITRE 7

PARADIGMES DE LA CONCURRENCE

Plan

COMPORTEMENT OPÉRATIONNEL ET CONCURRENCE DES PROCESSUS
ACTUALITÉ DE LA CONCURRENCE
PARADIGMES ET ARCHÉTYPES
ÉTUDES DE CAS
PROPRIÉTÉS DU COMPORTEMENT

Manuel 7

Paradigmes de la concurrence

1. Comportement opérationnel et concurrence des processus

L'étude du cahier des charges et l'élaboration des spécifications peuvent déjà dégager des formes de concurrence provenant de la répartition ou d'un parallélisme intrinsèque dû à l multiplicité des acteurs et des moyens qui interviennent dans l'application.

Mais c'est au niveau de l'analyse opérationnelle qu'on introduit le parallélisme d'exécution, celui qui est contrôlé par le système d'exploitation ou par la plate-forme support de l'architecture matérielle informatique.

Le comportement opérationnel de l'application est exprimé par le comportement concurrentiel des processus informatiques, ceux que nous avons introduits au chapitre 1.

2. Actualité de la concurrence

La concurrence entre processus est un sujet de grande actualité pour plusieurs raisons.

1. L'évolution des systèmes personnels : Dos, Windows, Mac, fait passer ceux-ci d'une organisation monoprocessus à des systèmes multiprogrammés.
2. Les systèmes classiques, MVS, AS 400 (IBM), VMS (Digital) , Solaris (Sun), Unix, Linux, qui sont multiprogrammés dès leur conception continuent à se développer et sont de plus en plus souvent multiprocesseurs.
3. Les applications temps réel, embarquées, enfouies, se développent et requièrent des systèmes temps réel multiprocessus (on dit, dans ce métier, des exécutifs multitâches)
4. Les applications réparties se multiplient et chacun peut désormais écrire ou utiliser des processus concurrents à distance avec des applets ou des servlets Java. Les systèmes d'architecture client-serveur, les architectures réparties à base d'objets ou de composants vont aussi dans le sens d'une augmentation du nombre des processus concurrents.
5. Enfin, les clients nomades, les plates-formes mobiles, ne peuvent fonctionner sans utiliser des processus mandataires ou proxys qui sont leurs représentants sur les serveurs fixes de ressource.

La connaissance des bons comportements et des bonnes méthodes de programmation et d'utilisation de ces processus concurrents est indispensable.

3. Paradigmes et archétypes

On appelle paradigmes de la concurrence des exemples type qui permettent de modéliser des classes de problèmes qui sont fréquemment rencontrés et qui sont présents à tous les niveaux dans les systèmes et dans les applications concurrentes. Ils sont acceptés par la communauté informatique pour leur capacité à rendre compte du réel et à servir de schémas de concurrence lors de l'analyse et la conception des systèmes ("concurrency design patterns").

Les solutions des paradigmes fournissent des schémas de conception et de programmation de composants concurrents ("concurrency components"). Elles servent de briques de base à toute étude, analyse ou construction de système ou d'application coopérative.

La solution d'un paradigme est un archétype qu'il faut connaître par coeur parce que c'est le modèle général d'une certaine structure qui détermine le comportement acceptable pour l'ensemble des processus concurrents dans le cadre du problème à résoudre, et cela quelle que soit la nature des processus. L'architecture des systèmes et des applications l'utilise soit directement, soit par composition, soit en dérivant de nouvelles solutions à partir du modèle de base.

4. Études de cas

Les principaux paradigmes de la concurrence que nous étudions sont :
- l'exclusion mutuelle qui modélise l'accès cohérent à de ressource partagées,
- la cohorte qui modélise la coopération d'un groupe de taille maximale donnée,
- le passage de témoin qui modélise la coopération par division du travail entre processus,
- le schéma producteurs-consommateurs qui modélise la communication par un canal fiable,
- le schéma lecteurs-rédacteurs qui modélise la compétition cohérente,
- le schéma du repas des philosophes qui modélise l'allocation de plusieurs ressources.

Par le paradigme de l'exclusion mutuelle, on exprime que les accès des processus concurrents à une ressource doivent préserver la cohérence des informations conservées par cette ressource; c'est une propriété de fiabilité ou de sûreté de fonctionnement (on doit garantir que rien de "mauvais" ne peut se produire, ici serait "mauvais" la perte de l'information par incohérence due à la concurrence).

Précisons un peu cette notion d'accès à une ressource matérielle ou logicielle (ce peut être un objet d'un langage à objet);

Si l'utilisation de la ressource peut être considéré comme une action unique, qui a lieu complètement ou pas du tout, qui est insécable ou indivisible, on dit qu'on a un accès atomique. Dans ce cas, qu'il y ait un demandeur d'accès ou plusieurs ne change rien, car l'accès, c'est à dire l'utilisation de la ressource, est le même dans tous les cas : il ne peut pas y avoir concurrence d'accès.

Si l'utilisation de la ressource nécessite une suite d'actions où chaque action correspond à un accès, on dit qu'on a une suite d'accès. Et c'est la nature de cette suite qui peut imposer des comportements particuliers à tout groupe de processus.

Distinguons les différentes utilisations possibles par un groupe de processus.

- Si, à tout moment, la ressource ne sert qu'à un seul processus du groupe, on dit qu'on a une utilisation isolée ; ceci peut se produire si un processus seulement connaît la ressource ou en a le droit d'accès;

- Si, au contraire, tous les processus du groupe ont le droit d'utiliser la ressource, on a une utilisation concurrente, et celle-ci peut intervenir selon plusieurs modalités :

*- l'utilisation est sérialisée si la ressource est allouée à un processus à la fois et si seul ce processus qui a obtenu cette allocation peut utiliser la ressource ; les processus demandeurs de ressource et non satisfaits sont mis en attente et sont servis l'un après l'autre ; un seul processus a le droit d'appeler la ressource (pour un objet, c'est le droit d'appeler une méthode de l'objet) ;

*- l'utilisation est exclusive si tous les processus ont le droit d'appeler la ressource (pour un objet, tous les processus ont le droit d'appeler une méthode de l'objet), mais seul un processus peut l'utiliser ; c'est très voisin de l'utilisation sérialisée : l'attente se fait dans la ressource d'utilisation exclusive alors qu'elle se fait à l'extérieur d'une ressource d'utilisation sérialisée; dans les deux cas, il n'y a qu'un processus qui déroule la suite des accès à la ressource ;

*- l'utilisation est parallèle si plusieurs processus peuvent interclasser les accès de leurs suites respectives d'accès ;

*- l'utilisation est parallèle plafonnée si le nombre de processus qui peuvent utiliser ensemble la ressource est borné.

Les modalités de concurrence dépendent de la nature de la suite d'accès de chaque processus. Si l'utilisation de la ressource consiste en une suite de lectures, il faut que tous les accès de la suite correspondent à la même version de la ressource, c'est à dire qu'il ne faut pas qu'il y ait d'écriture concomitante à cette suite de lecture, donc des utilisations avec lectures seules peuvent être parallèles, plafonnées ou non; mais toute utilisation comportant une écriture doit être exclusive avec toute utilisation avec lectures seules ou avec toute autre utilisation avec écriture.

On peut respecter ces contraintes en fonctionnant en exclusion mutuelle entre les processus utilisateurs, mais aussi avec le schéma lecteurs-rédacteurs (voir plus loin).

Par le paradigme de la cohorte, on exprime la gestion coopérative d'un lot de ressources banalisées ou le contrôle de la concurrence dans un serveur multiprocessus.

Par le paradigme du passage de témoin, on exprime la coopération par division du travail entre processus. On y trouve plusieurs cas : des processus travaillent à la chaîne ; un processus envoie un signal à son successeur, en passant aussi le droit d'accès à une ressource commune ; rendez-vous entre processus ; appel procédural local ou distant.

Par le paradigme producteurs-consommateurs, on exprime une coopération fiable car on réalise un contrôle de flux et on garantit que tout message émis est bien reçu une fois et une seule, après avoir été émis.

Par le paradigme lecteurs-rédacteurs, on exprime que plusieurs processus lecteurs peuvent utiliser la ressource en parallèle, soit parce que leur utilisation ne change pas l'état de la ressource, soit parce que chaque processus a accès à une sous-ressource indépendante des autres (chaque accès participe à une utilisation isolée de chaque sous-ressource). Mais les rédacteurs sont exclusifs entre eux ou avec un lecteur.

Avec le paradigme du repas des philosophes, on étudie la sûreté de l'allocation de deux ressources pour garantir l'absence d'interblocage et on étudie la vivacité pour garantir l'équité de l'allocation et éviter la famine d'un des processus.

5. Propriétés du comportement de l'ensemble des processus

Les processus concurrents que l'on étudie ici ont un comportement asynchrone. C'est à dire qu'ils ont chacun un programme séparé et qu'ils l'exécutent soit en multiprogrammant une unité centrale, soit en ayant chacun un processeur. Ces processeurs ont chacun leur propre cycle d'instructions. Ils partagent éventuellement une horloge qui est alors celle du système d'exploitation. Ils peuvent aussi ne communiquer que par messages, ce qui est le cas des systèmes répartis.

Que les processus concurrents soient centralisés (c'est à dire partageant une mémoire centrale commune) ou répartis (c'est à dire communiquant uniquement par messages), ils doivent présenter des comportements corrects, c'est à dire sûrs et vivaces. La sûreté est la propriété qui garantit que rien de "mauvais" ne peut se produire (que le ressource ou les messages sont cohérents et non écrasés ou surchargés). La vivacité est la propriété qui garantit que quelque chose de "bon" finira par se produire (que l'utilisation de la ressource ou du canal est équitable et qu'on évite la famine d'un des processus).