Ordonnancement

Objectifs du TP :

1. Ordonnancement `CPU` (*Operating System Concepts*, Silverschatz, Galvin, Gagne) ;
2. Ordonnancement multiprocesseur (*cf.* poly d'Algorithmique Parallèle).

1 Ordonnanceur système

Question. Décrivez les actions prise par un noyau pour changer de contexte entre deux threads noyaux.

Question. On suppose que les processus suivants arrivent prêts à être ordonnancés aux temps indiqués. Chaque processus a une durée d'exécution connue indiquée. Les décisions d'ordonnancement sont prises en fonction des informations connues au moment de la prise de décision.

Processus	Date d'arrivée	Durée d'exécution
P1	0.0	8
P2	0.4	4
P3	1.0	1

- Quel est le temps moyen de terminaison (date de fin d'exécution moins date d'arrivée) d'un processus avec un algorithme `FCFS` (First Come First Serve) ?
- Avec un algorithme `SJF` (Shortest Job First) ?
- Avec un algorithme `SJF`, en ne faisant rien (idle) la première seconde.

Question. La plupart des algorithmes d'ordonnancement système sont paramétrés. Quel(s) paramètre(s) utilise par exemple l'algorithme Round Robin ? Et l'algorithme MultiLevel Feedback Queues ? Ces algorithmes sont donc des ensembles d'algorithmes ou classes. Une classe peut en inclure une autre. Donnez les relations entre les classes liées aux algorithmes :

- `FCFS` et `RR` ;
- Priorité et `SJF` ;
- `MLFQ` et `FCFS` ;
- Priorité et `FCFS` ;
- `RR` et `SJF`.

Question. On suppose qu'un algorithme d'ordonnancement (ordonnancement CPU à court-terme) favorise les processus qui ont utilisé le moins le processeur. Les programmes bornés par les I/Os seront-ils favorisés ? Quel est l'impact sur les programmes bornés par l'utilisation du CPU ?

Question. Un système peut-il détecter que certains de ses processus sont en famine ? Expliquez ?

2 Ordonnancement multiprocesseur d'un graphe FORK avec communications multi-port

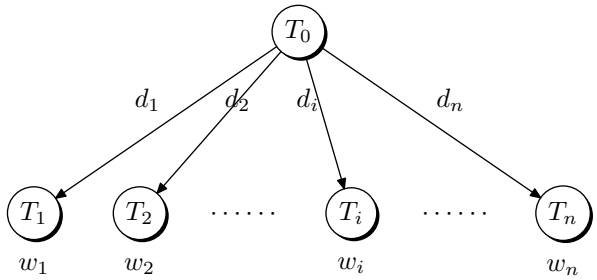

FIG. 1 – Graphe de FORK a n fils.

Définition. `FORK à` n `fils.` *Un graphe FORK à n fils est un graphe de tâches à $n+1$ nœuds étiquetés par $T_0, T_1, \cdots, T_n$, comme illustré figure 1. Il y a une arête entre le nœud T_0 et chacun de ses fils T_i, $1 \leqslant i \leqslant n$. Chaque nœud possède un poids w_i qui représente le temps de traitement de la tâche T_i. Chaque arête (T_0, T_i) possède aussi un poids correspondant au volume de données échangées d_i si la tâche T_0 et la tâche T_i ne sont pas traitées sur le même processeur.*

On suppose d'abord disposer d'une infinité de processeurs identiques et multi-port (qui peuvent initier plusieurs envois simultanés). On définit le problème d'optimisation suivant :

DefinitionFORK-SCHED-$\infty(G)$ Étant donné un graphe FORK G à n fils et un ensemble infini de processeurs identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?

Question. Donner un algorithme polynomial résolvant FORK-SCHED-∞.

On s'intéresse maintenant au même problème avec un nombre borné de processeurs :

Définition. `FORK-SCHED-BOUNDED(`G`,`p`).` *Étant donné un graphe FORK G à n fils et un ensemble de p processeurs identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?*

Question. Montrer que le problème de décision associé à FORK-SCHED-BOUNDED est NP-complet.

On revient enfin au problème avec une infinité de processeurs identiques, mais on suppose désormais qu'un processeur ne peut communiquer qu'avec un seul processeur à la fois (modèle 1-port).

Définition. `FORK-SCHED-1-PORT-`∞`(`G`).` *Étant donné un graphe FORK G à n fils et un ensemble infini de processeurs 1-port identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?*

Question. Quid de FORK-SCHED-1-PORT-∞ ? (On demande une intuition. La preuve sera vue dans le cadre du cours d'AlgoPar).