TP1 : Etude de l'ordonnancement de processus par simulation

RSA : Réseaux et Systèmes Avancés
Deuxième année

Nous allons étudier lors de cette séance le comportement de différents algorithmes d'ordonnancement à l'aide du simulateur *Process Scheduling* développé à l'Université du Texas et disponible en ligne à l'adresse : `http://vip.cs.utsa.edu/simulators`. Ce simulateur nous permettra de comparer les performances des algorithmes d'ordonnancement et d'évaluer l'impact des différents paramètres de simulation. **Vous rédigerez un rapport de TP de cinq pages maximum que vous rendrez en fin de séance à votre encadrant.** Dans ce rapport, vous indiquerez à chaque fois les expériences réalisées, justifierez le choix de vos paramètres et donnerez une part importante à l'analyse des résultats obtenus.

1 Installation du simulateur *Process Scheduling*

▷ **Question 1.** Téléchargez et décompressez l'archive zip (qui contient le simulateur codé en Java, les fichiers de configuration et les bibliothèques requises) depuis l'adresse : `http://vip.cs.utsa.edu/simulators/zipfiles/ps.zip`

▷ **Question 2.** Lancez l'application à l'aide du script `runps` (sous Linux) ou `runps.bat` (sous Windows) depuis le répertoire d'installation, et vérifiez que le programme fonctionne correctement en lançant une expérimentation ("Run experiment") et en traçant les temps de réponse ("Graph type : Turnaround" + "Draw"). Quittez l'application.

2 Configuration des scénarios de simulation

Ce simulateur va nous permettre de définir et exécuter différents scénarios de simulation. Un scénario (*experiment run*) est défini par un algorithme d'ordonnancement et un ensemble de processus à exécuter selon cet algorithme. Une expérimentation (*experiment*) consiste en un ensemble de scénarios qui sont comparés et analysés à l'aide du simulateur. Le simulateur est livré avec deux fichiers de configuration par défaut : `myexp.exp` et `myrun.run` qui permettent de définir les scénarios de simulation.

▷ **Question 3.** En vous aidant de la documentation en ligne disponible à l'adresse : `http://vip.cs.utsa.edu/simulators/guides/ps/ps_doc.html`, déterminez le rôle de chacun des fichiers `myexp.exp` et `myrun.run` ainsi que leurs relations. Combien de scénarios sont décrits par les fichiers par défaut ? Lesquels ?

▷ **Question 4.** De la même façon, déterminez à quoi correspondent les paramètres suivants dans le fichier `myrun.run` : `algorithm`, `numprocs`, `firstarrival`, `interarrival`, `duration`, `cpuburst` et `ioburst` ? Quelle est la différence entre le paramètre `duration` et le paramètre `cpuburst` ?

3 Génération des résultats de simulation

▷ **Question 5.** Tracez le diagramme de Gannt correspondant au scénario par défaut implantant l'ordonnancement FCFS (*First-Come First-Serve*). Quels sont les temps de réponse minimal et maximal observables ? Retrouvez ces valeurs à partir des données du tableau résultat.

▷ **Question 6.** Tracez le graphique présentant les temps de réponse. A quoi correspondent précisément les abscisses et ordonnées du graphique ?

4 Performances de l'algorithme d'ordonnancement RR

▷ **Question 7.** Réalisez une ou plusieurs expérimentation(s) permettant de vérifier ou de réfuter l'hypothèse suivante : « L'algorithme d'ordonnancement RR (Round-Robin) appliqué dans un scénario avec n processus donne le sentiment à l'utilisateur que la machine fonctionne au (1/n)-ième de sa vitesse tant que les temps consacrés aux opérations d'entrée-sortie restent faibles.»

▷ **Question 8.** Etablissez une relation entre le temps de réponse maximal, la durée des processus et le nombre de processus pour un temps I/O faible.

▷ **Question 9.** Déterminez quantitativement à partir de quelle valeur le temps consacré aux I/O peut être considéré comme suffisamment faible dans notre contexte.

5 Comparaison des ordonnancements FCFS et SJF

▷ **Question 10.** Réalisez une ou plusieurs expérimentation(s) permettant de comparer les performances des algorithmes d'ordonnancement FCFS (*First-Come First-Serve*) et SJF (*Shortest-Job-First*). Quel critère de performances peut être amélioré grâce à l'algorithme SJF ?

▷ **Question 11.** Identifiez et exécutez une ou plusieurs expérimentation(s) où ces deux algorithmes d'ordonnancement FCFS et SJF offrent les mêmes performances. Justifiez votre réponse.

6 Impact de la préemption sur l'ordonnancement SJF

Nous considérons 15 processus arrivant au même instant et d'une durée identique de 300 UT. Les processus ont un temps CPU uniformément distribués entre 10 et 20 UT et un temps I/O uniformément distribué entre 100 et 200 UT.

▷ **Question 12.** Réalisez une expérimentation utilisant ces processus et permettant de comparer les algorithmes d'ordonnancement SJF et PSJF (SJF avec préemption). En quoi consiste la préemption ? Observez-vous une différence entre les deux scénarios ? Expliquez pourquoi.

▷ **Question 13.** Calculez le *load average* (charge moyenne) pour chacun de ces deux scénarios. Le *load average* désigne sous les systèmes UNIX une mesure de la quantité de travail du système qui correspond au nombre moyen de processus dans l'état prêt (*ready*). Il peut typiquement être obtenu à l'aide de la commande `top` ou `uptime`. Cette information n'est pas directement fournie par le simulateur mais vous pouvez la calculer à partir des données du tableau résultat qu'il génère.

7 Changement de contexte (*context switch*)

Le simulateur ne prend pas en compte les temps nécessaires pour changer de contexte (*context switch time*). Il est possible d'intégrer ce paramètre en procédant comme suit : chaque fois qu'un changement de contexte a lieu, un temps doit être ajouté au temps d'attente de chaque processus qui est dans l'état prêt (*ready*).

▷ **Question 14.** Reprenez les scénarios de simulation par défaut (FCFS et SJF) et déterminez à partir des tableaux résultats le nombre de changements de contexte pour chacun des deux scénarios.

▷ **Question 15.** Déterminez, par le calcul, pour quelle valeur la durée d'un changement de contexte provoque une augmentation de 10% du temps d'attente moyen pour chacun des scénarios de simulation.