

Projet de validation du module de recherche opérationnelle

Chaque groupe doit rendre un projet correspondant à son propre travail.

Ce projet illustre directement le cours sur les files d'attente donné dans le cadre du module de Recherche Opérationnelle. L'énoncé contient néanmoins suffisamment d'informations pour les élèves qui n'auraient pas assisté au cours de R.O. Ainsi, les formules théoriques utiles sont rappelées à la fin de l'énoncé du projet.

But : Simulation du comportement d'un système informatique pour améliorer ses performances

Sujet : Système d'attente classique M/M/m.

Considérons un système informatique composé de plusieurs processeurs fonctionnant en parallèle. Pour étudier a priori les performances (temps de réponse, débit, équilibrage de la configuration) d'un tel système, il est souvent intéressant de modéliser son comportement.

Le modèle que nous utiliserons est constitué d'une file d'attente dans laquelle patientent les travaux à traiter, et de m serveurs qui sont les processeurs, l'ensemble formant ce qu'on appelle par définition un système d'attente. Précisons le fonctionnement : la file d'attente est unique ; dès qu'un travail a été traité par un processeur, celui qui est en tête de la file d'attente le remplace immédiatement (ou si la file est vide, le processeur devient inactif). Les travaux se rangent dans la file dans l'ordre d'arrivée. Ils arrivent à des instants indépendants (processus de Poisson), leur nombre moyen par seconde étant de $\lambda = 4 \text{ s}^{-1}$. La durée de traitement d'un travail par un processeur suit une loi exponentielle de taux $\mu = 5 \text{ s}^{-1}$, c'est à dire qu'en moyenne 5 travaux sont traités par seconde (attention λ et μ ne sont pas nécessairement des valeurs entières). La notation de Kendall de ce phénomène d'attente est donc M/M/m.

L'ordinateur doit traiter $M = 1000$ travaux (pour les tests il devra pouvoir en tester beaucoup plus, par exemple 1 000 000).

On appellera : N_Q : le nombre de travaux dans la file d'attente
 N : le nombre de travaux dans le système d'attente
 W_Q : le temps d'attente d'un travail dans la file d'attente
 W : le temps d'attente d'un travail dans le système d'attente

Question 1 :

Le système ne dispose, pour l'instant, que d'un processeur (système M/M/1).

1) Pour tester la génération des durées aléatoires de traitement des travaux, tracer l'histogramme des durées de traitement de ces travaux avec des axes semi-logarithmiques (ordonnée : $\ln(\text{nombre de travaux de durée comprise entre } x \text{ et } x+\Delta x)$; abscisse : durée x). On doit obtenir une droite dont la pente permet de retrouver la valeur de μ .

remarque : cette première question est capitale pour la suite du projet, inutile d'aller plus loin tant que les tests demandés ne sont pas concluants. Pour ce test et les questions suivantes on pourra être amené à traiter un plus grand nombre de travaux pour s'affranchir du fait que les résultats trouvés au cours de tests successifs présentent un caractère fortement aléatoire lorsque l'on traite seulement 1000 travaux.

2) Calculer et tracer $N_Q(t)$. On mettra en évidence le régime transitoire, suivi du régime permanent.

3) Calculer, en régime permanent, la valeur moyenne des quatre paramètres $\bar{N}_Q$, $\bar{N}$, $\bar{W}_Q$, $\bar{W}$, que l'on comparera aux résultats théoriques du cours. On donnera également l'écart-type de ces paramètres, ainsi que la précision exprimée par le rapport "écart-type / valeur moyenne".

4) Calculer, en régime permanent, le taux d'utilisation q^* du processeur, le débit D du système, c'est à dire le nombre moyen de travaux traités par seconde, et le temps total pour traiter tous les travaux. Discuter ces résultats.

Question 2 :

Pour améliorer les performances du système, on rajoute un deuxième, puis un troisième processeur identiques au premier. Pour $m = 2$, puis $m = 3$:

1) Calculer et tracer $N_Q(t)$.

2) Calculer, en régime permanent, la valeur moyenne des quatre paramètres $\bar{N}_Q$, $\bar{N}$, $\bar{W}_Q$, $\bar{W}$. Comparer le temps moyen $\bar{W}_Q$ au résultat théorique du cours. On notera que le fait d'ajouter un deuxième processeur diminue ce temps moyen d'un facteur largement supérieur à 2.

3) Calculer, en régime permanent, le taux d'utilisation q^* de chaque processeur, le débit D du système, et le temps total pour traiter tous les travaux. Conclusion.

Remarques :

Les données du problème seront introduites dans un fichier de paramètres dont le format est le suivant :

1ère ligne :	P;4.0	Poisson_lambda(1/s)_arrivée
2ème ligne :	E;5.0	Exponentielle_mu(1/s)_durée
3ème ligne :	1;	nombre_de_processeurs
4ème ligne :	1000;	nombre_de_travaux_à_traiter

A chaque question correspond un fichier de paramètres ; leurs noms seront "para1.p", "para2.p".

Le logiciel sera écrit en C ou C++ et l'exécutable devra pouvoir tourner sur un PC standard sous Windows XP.

Afin de pouvoir tester un autre fichier de paramètres, on devra pouvoir passer en premier argument, le nom du fichier de paramètres.

Les résultats sur ordinateur devront être présentés sous forme de tableaux puis les courbes seront tracées sur papier.

Travail à remettre :

- le travail sera effectué par groupe de 3 personnes faisant **impérativement** partie du même groupe de TD : les délégués seront chargés de communiquer ces groupes aux chargés de TD ainsi qu'à l'administration de l'EFREI.
- un rapport écrit exposant clairement vos résultats (20 pages maximum).
- un exposé oral sera préparé. Sa durée, l'ordre de passage ainsi que la salle de soutenance vous seront communiqués ultérieurement. Moyens pour la présentation : tableau, rétroprojecteur ou vidéoprojecteur. Si vous choisissez le vidéoprojecteur (moyen recommandé), les présentations devront être préalablement chargées sur une clé USB au format MS OFFICE2003.
- un logiciel (écrit en C ou C++ et tournant sous Windows) sera réalisé pour répondre aux questions ; afin de tester ce logiciel pendant la présentation, le logiciel devra lire les paramètres du problème d'affectation dans un fichier de configuration dont la structure est indiquée dans l'énoncé.

Date de remise : le vendredi 29 mai 2009 au plus tard à l'accueil des enseignants.

(en cas de problème, me contacter par e. mail : vincent.vivier@upmc.fr)

Génération de nombres aléatoires

Génération de nombres aléatoires de densité donnée

Cas général : utiliser la méthode ITM "inverse transformation method", voir les photocopies jointes.

Cas d'une densité exponentielle de paramètre λ : méthode ITM

- tirer au hasard un nombre aléatoire x de loi uniforme entre 0 et 1
- le nombre $y = -\frac{1}{\lambda} \ln(1 - x)$ obéit à une loi de densité

exponentielle de paramètre λ .

Génération d'un processus de Poisson

Si les instants aléatoires T_i où se produisent des événements sont indépendants, les intervalles de temps W_i entre deux événements successifs suivent une loi de densité exponentielle de paramètre λ , où λ est le nombre moyen d'événements par seconde.

Ainsi, pour générer les instants $T_0 = 0, T_1 = T_0 + W_0, T_2 = T_1 + W_1, \dots$ il suffit de savoir générer les intervalles de temps W_i obéissant à une loi exponentielle (voir ci-dessus)

Files d'attente : valeurs moyennes des quantités N, N_Q, W, W_Q

Système d'attente M/M/1 : 1 serveur

$$\overline{N} = \frac{\lambda}{\mu - \lambda} \quad \overline{N}_Q = \frac{\lambda^2}{\mu(\mu - \lambda)} \quad \overline{W} = \frac{1}{\mu - \lambda} \quad \overline{W}_Q = \frac{\lambda}{\mu(\mu - \lambda)}$$

Système d'attente M/M/m : m serveurs

$$\overline{W}_Q = \frac{\left(\frac{\lambda}{\mu}\right)^m}{\mu m m! \left(1 - \frac{\lambda}{m\mu}\right)^2} p_0 \quad \text{avec} \quad p_0 = \frac{1}{\sum_{n=0}^{m-1} \frac{1}{n!} \left(\frac{\lambda}{\mu}\right)^n + \left(\frac{\lambda}{\mu}\right)^m \frac{1}{m!} \frac{1}{1 - \frac{\lambda}{m\mu}}}$$

$$\overline{W} = \overline{W}_Q + \frac{1}{\mu} \quad \overline{N}_Q = \lambda \overline{W}_Q \quad \overline{N} = \lambda \overline{W}$$

7.2 Transformation Method: Exponential and Normal Deviates

In the previous section, we learned how to generate random deviates with a uniform probability distribution, so that the probability of generating a number between x and $x + dx$, denoted $p(x)dx$, is given by

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases} \quad (7.2.1)$$

The probability distribution $p(x)$ is of course normalized, so that

$$\int_{-\infty}^{\infty} p(x)dx = 1 \quad (7.2.2)$$

Now suppose that we generate a uniform deviate x and then take some prescribed function of it, $y(x)$. The probability distribution of y , denoted $p(y)dy$, is determined by the fundamental transformation law of probabilities, which is simply

$$|p(y)dy| = |p(x)dx| \quad (7.2.3)$$

or

$$p(y) = p(x) \left| \frac{dx}{dy} \right| \quad (7.2.4)$$

Exponential Deviates

As an example, suppose that $y(x) \equiv -\ln(x)$, and that $p(x)$ is as given by equation (7.2.1) for a uniform deviate. Then

$$p(y)dy = \left| \frac{dx}{dy} \right| dy = e^{-y} dy \quad (7.2.5)$$

which is distributed exponentially. This exponential distribution occurs frequently in real problems, usually as the distribution of waiting times between independent Poisson-random events, for example the radioactive decay of nuclei. You can also easily see (from 7.2.4) that the quantity y/λ has the probability distribution $\lambda e^{-\lambda y}$.

So we have

```
#include <math.h>

float expdev(long *idum)
Returns an exponentially distributed, positive, random deviate of unit mean, using
ran1(idum) as the source of uniform deviates.
{
    float ran1(long *idum);
    float dum;

    do
        dum=ran1(idum);
    while (dum == 0.0);
    return -log(dum);
}
```