

Antoine Acquaviva Matthias Meusburger Groupe 1A	IUP GMI
---	---------

Modélisation des systèmes :

Simulation d'une file d'attente M/M/1

SOMMAIRE

I) Problématique.	3
II) Méthode.	5
III) Application Finale.	8

Nota Bene : Etant donné la modeste taille de ce rapport, et dans un souci de clarté, les introductions et conclusions sont inhérentes respectivement à la première et à la dernière partie.

I. LA PROBLEMATIQUE

I.1. Les files d'attente

La théorie des files d'attente permet de modéliser les systèmes dans lesquels un client attend un service. Il existe de nombreux exemples de files d'attente : caisses de supermarché, attente à un guichet... En informatique cette théorie s'applique aux processus en attente de traitement.

Les différents paramètres caractérisant une file d'attente sont :

- A : Loi de l'intervalle de temps entre deux arrivées
- B : Loi du temps de service
- C : Nombre de serveurs
- K : Nombre maximum de clients admis dans le système
- m : Nombre de clients potentiels
- Z : Discipline de la file

I.2. Les files d'attente de type MM1

Les files MM1 sont un cas particulier de files d'attente, c'est ce type de file qui nous intéresse ici. Les particularités d'un tel système sont :

- La loi des temps inter-arrivées (A) est la loi exponentielle de paramètre λ .
- La loi des temps de services (M) est la loi exponentielle de paramètre μ .
- Le nombre de serveurs (c) est réduit à un.
- La discipline de la queue est FIFO.
- On ne limite pas le nombre de clients dans le système ($K = \infty$).

Les résultats prévus par la théorie pour les systèmes MM1 sont :

- Nombre moyen de clients dans le système :

$$L = E(N) = \frac{\rho}{1-\rho} \text{ avec } \rho = \frac{\lambda}{\mu}$$

- Temps de service moyen : $E(s) = \frac{1}{\mu}$
- Temps moyen passé par le client dans le système $W = \frac{1}{\mu \times (1-\rho)}$
- Temps moyen passé dans la queue $W_q = \frac{\rho}{1-\rho} \times E(s)$
- Nombre moyen de clients dans la queue : $L_q = \frac{\rho^2}{1-\rho}$

I.3. Automatisation des simulations

Pour valider le modèle il convient d'effectuer de nombreuses simulations. Or, une simulation de file d'attente MM1 nécessite de nombreux calculs pour déterminer les temps inter-arrivées et de service et en déduire les diverses valeurs prévues par la théorie. Pour effectuer ces calculs de manière infaillible et reproductible il faut automatiser ces simulations. Le simulateur ainsi constitué devra être en mesure d'effectuer une simulation pour n'importe quel triplet de valeurs de λ , μ et p (p étant le nombre de processus).

II. LA METHODE

II.1. Pourquoi un programme ?

Un programme informatique est la moyen le plus adapté pour réaliser ce simulateur. Il permet de répéter l'expérience un grand nombre de fois en un temps raisonnable. Un tel programme peut prendre en entrée les paramètres du simulateur et fournir en sortie les valeurs déterminées par la simulation. Il permet aussi d'automatiser la production de graphiques permettant une interprétation plus aisée des résultats de la simulation. Enfin, il permet d'enregistrer sous forme de fichier les résultats d'une simulation.

II.2. Cahier des charges

L'application doit permettre deux catégories de simulations :

- Simuler l'évolution d'une file d'attente MM1 composés de p processus dont les lois sont de paramètres λ et μ (premier scénario).
- Simuler l'évolution de n files MM1, chacune composée du même nombre de processus tous régis par les lois de même paramètres (deuxième scénario).

En entrée, l'application prend les valeurs de n , p , λ , μ déterminées par l'utilisateur. Puis la simulation commence, dans une première étape, à chaque processus est associée ses temps inter-arrivées et de service, dans une deuxième étape les temps d'attente sont déterminés. Au terme de ces deux étapes, deux fichiers sont enregistrés, l'un contenant les trois temps associés à chaque processus, le deuxième comprenant les dates correspondantes en prenant comme date 0 l'arrivée du premier processus dans le système.

Ensuite, à la demande de l'utilisateur, le programme doit tracer les graphiques représentant ces temps (inter-arrivée, traitement, service) par processus.

Finalement, le programme doit afficher une page d'analyse de la simulation contenant les valeurs théoriques et les valeurs déterminées par la simulation de ces temps moyens, une évaluation de la charge théorique et pratique du système.

II.3. Choix techniques

Nous avons choisi, pour développer cette application, d'utiliser le langage Java. En effet, bien qu'interprété, et par là-même relativement lent, Java est aussi multi-plateformes, et doit donc pouvoir s'exécuter dans n'importe quel environnement muni d'une machine virtuelle. Cet aspect « universel » nous a séduit, et nous avons choisi de faire un compromis sur la rapidité d'exécution. Par ailleurs, la complexité des calculs étant plutôt faible (de l'ordre de n , avec n : nombre total de processus gérés), il nous a semblé que ce compromis ne serait pas ressenti au niveau de l'utilisation de l'application.

De plus, Java permet de réaliser des interfaces graphiques performantes de manière relativement aisée, et ce, grâce à des composants prédéfinis. Deux composants graphiques existent dans Java : Awt et Swing. En dépit de sa rapidité moindre, nous avons choisi d'utiliser Swing, étant donné la richesse des éléments qu'il propose. Swing ayant été implémenté à partir de la version 1.0 de Java, l'application devrait pouvoir fonctionner sur toute machine virtuelle récente.

Finalement, nous avons développé ce programme conjointement sous deux systèmes d'exploitations, Windows (NT4) et Linux

(Mandrake 8.1), afin de nous assurer de son bon fonctionnement sous ces deux environnements.

III. APPLICATION FINALE

III.1. Description des sources de l'application

L'application ayant été développée avec un langage objet, notre programme se décompose en différentes classes, correspondant aux différents concepts manipulés, et que l'on peut diviser en deux catégories. Tout d'abord, voici les classes qui servent à la simulation en elle-même :

- **Simulation** permet d'effectuer une simulation MM1.
- **NSimulations** permet d'effectuer n simulations MM1.
- **Resultats** permet d'afficher les valeurs issues de l'exécution des classes Simulation ou NSimulations.
- **Graphe** permet de tracer un graphique représentant les valeurs issues d'une ou plusieurs simulations.
- **Analyse** permet d'analyser les résultats obtenus après une simulation.

Ensuite, certaines classes servent directement à l'application :

La classe **Fenetre** est la plus importante. C'est dans cette classe que l'interface graphique sera construite, et que les actions de l'utilisateur sur cette interface seront traitées.

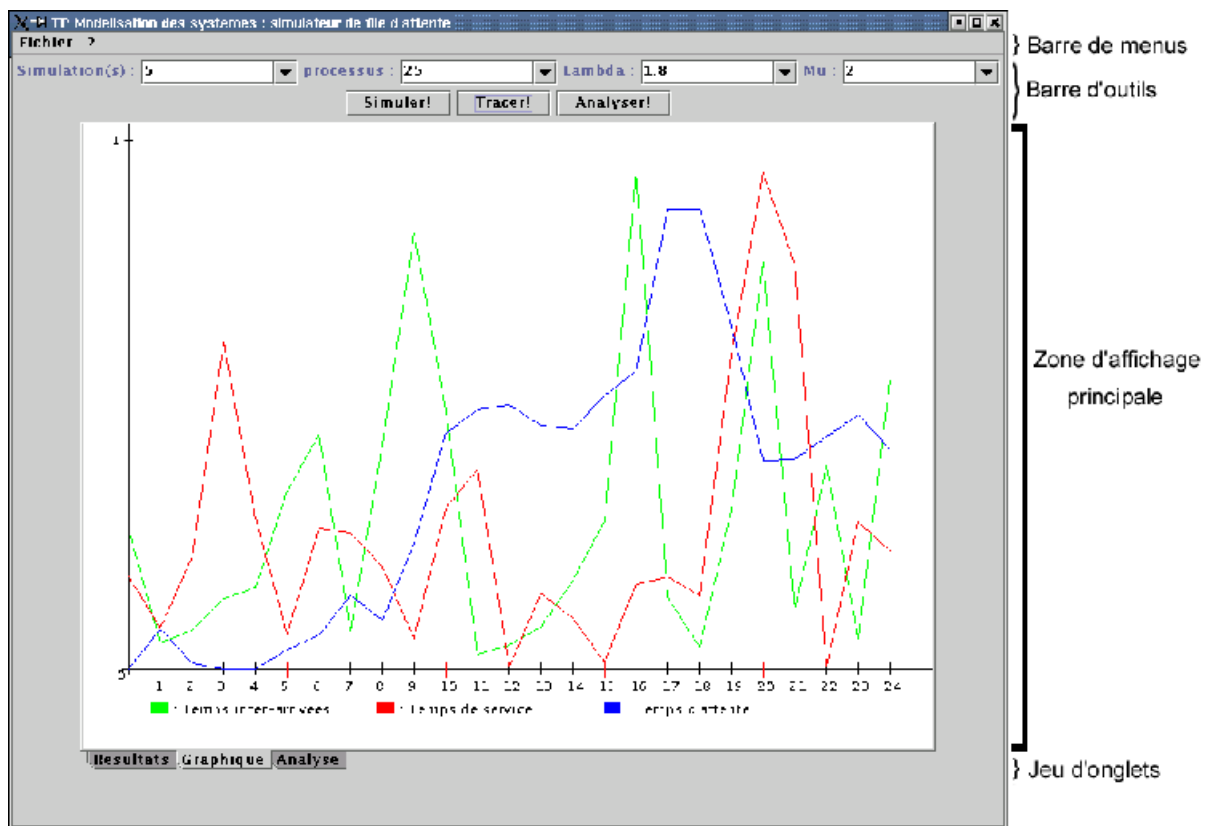
La classe **ModelePourTable** est une classe qui sert à définir un modèle pour l'affichage des valeurs numériques issues d'une simulation. Elle servira donc à la classe Resultats.

Finalement, la classe **AppliMS** contient le programme principal. Cette classe sert uniquement à instancier un objet de type Fenetre et à le rendre visible.

III.2. Utilisation

L'interface graphique de l'application se compose de quatre parties. Une barre de menus permet à l'utilisateur d'enregistrer les valeurs issues de la simulation dans des fichiers autres que ceux par défaut, grâce à une boîte de dialogue contenant l'arborescence du système de fichier courant. Il peut également afficher une aide sommaire, ou quitter l'application.

Ensuite, une barre d'outils est consacrée au paramétrage de la simulation. En dessous de cette barre d'outils se situe la zone d'affichage principale. Et finalement, en dessous de cette fenêtre se situe un jeu d'onglets permettant de naviguer entre les différentes représentations de la simulation.



Une fois l'application lancée, l'utilisateur peut configurer la simulation à effectuer. Il dispose pour cela de plusieurs boîtes combo lui

permettant de modifier le nombre de simulations, le nombre de processus par simulation, ainsi que les paramètres de simulation qui sont λ et μ . Ces boites combo contiennent une liste déroulante de valeurs prédéfinies, mais l'utilisateur peut également y saisir une valeur quelconque.

Cette barre d'outils dispose également de trois boutons permettant d'effectuer trois actions distinctes, à savoir : lancer une simulation et afficher les valeurs, tracer le graphique correspondant à la simulation courante, et enfin analyser les résultats obtenus. Ainsi l'utilisateur peut choisir ou non d'effectuer les deux dernières opérations. Il est à noter qu'une fois une simulation lancée, les valeurs obtenues sont enregistrés automatiquement dans deux fichiers, nommés par défaut "dates.txt" et "temps.txt". Ils seront écrasés et remplacés si une nouvelle simulation est lancée. Dans le cas où le nombre de simulations est supérieur à un, les fichiers s'appelleront "moyennesTemps.txt" et "moyennesDates.txt"

Finalement, un jeu d'onglets situé en dessous de la zone graphique permet de jongler entre l'affichage des trois étapes.

III.3. Conclusion sur les résultats

Après avoir testé notre simulateur pour de nombreuses combinaisons de paramètres différents, nous constatons que les valeurs déterminées par la simulation concordent avec celles prévues par la théorie lorsque le système est en sous-charge, la théorie ne prévoyant pas ces valeurs pour un système en surcharge.

Ces valeurs sont d'autant plus corrélées que le nombre de processus est grand, ce qui était prévisible puisque le caractère aléatoire s'atténue avec le nombre de processus.