UNIVERSITE DE NANCY I
U. E. R. de SCIENCES MATHEMATIQUES

ETUDE D'UN MODELE STOCHASTIQUE DE SYSTEMES D'EXPLOITATION

THESE DE 3e CYCLE

PRESENTEE PAR

geneviève JOMIER

le 4 juillet 1973

JURY : PRESIDENT Mr M. DEPAIX

EXAMINATEURS Mr C. PAIR

Mr J. C. DERNIAME

UNIVERSITE DE NANCY I
U. E. R. de SCIENCES MATHEMATIQUES

ETUDE D'UN MODELE STOCHASTIQUE DE SYSTEMES D'EXPLOITATION

THESE DE 3e CYCLE

PRESENTEE PAR

geneviève JOMIER

le 4 juillet 1973

JURY : PRESIDENT Mr M. DEPAIX

EXAMINATEURS Mr C. PAIR

Mr J. C. DERNIAME

UNIVERSITE DE NANCY I

ETUDE D'UN MODELE STOCHASTIQUE DE SYSTEMES D'EXPLOITATION

THESE DE 3e CYCLE

PRESENTEE PAR

geneviève JOMIER

le 4 juillet 1973

JURY: PRESIDENT Mr M. DEPAIX

EXAMINATEURS Mr C. PAIR

Mr J.C. DERNIAME

Je tiens à remercier :

Monsieur DEPAIX, Professeur à l'Université de Nancy I, qui a inspiré et dirigé cette recherche.

Monsieur PAIR, Directeur de l'Institut Universitaire de Calcul Automatique, qui me fait l'honneur de participer au jury.

Monsieur DERNIAME, pour l'intérêt qu'il a porté à ce travail, son aide et ses conseils.

Je souhaite enfin remercier tous les membres de l'Institut Universitaire de Calcul Automatique pour leur collaboration sympathique et tout particulièrement le secrétariat qui s'est chargé de la réalisation matérielle de cette thèse.

TABLE DES MATIERES

INTRODUCTION.

CHAPITRE I - Présentation des travaux existants sur l'évaluation des performances, l'optimisation et la comparaison des systèmes d'exploitation.

I-1. Introduction.

I-2. Les mesures.

I-2.1. Les mesures câblées.
I-2.2. Les mesures programmées.

I-2.2.1. Les mesures locales au système.

I-2.2.1.1. Mesures sur le matériel : les performances d'unités centrales.

I-2.2.1.1.1. Les assortiments d'instructions.
I-2.2.1.1.2. Les noyaux.

I-2.2.1.2. Mesures de programmes.

I-2.2.1.2.1. Méthode par déroutements.
I-2.2.1.2.2. La trace.

I-2.2.1.3. Etude du comportement des utilisateurs.

I-2.2.2. Les mesures globales.

I-2.2.2.1. Les mesures de type "comptabilité".

I-2.2.2.2. Les mesures par l'échantillonnage.

I-3. Les modèles analytiques.

I-3.1. Les modèles locaux.

I-3.1.1. Les modèles de gestion de l'ensemble des travaux.

I-3.1.2. Les modèles de gestion de la mémoire.

I-3.1.3. Les modèles de gestion des disques.

I-3.2. Les modèles globaux.

I-4. La simulation des systèmes.

I-5. Conclusion.

CHAPITRE II - Description d'un système de multiprogrammation.

II-1. Introduction.

II-2. Description du matériel.

II-2.1. L'unité centrale ou unité de calcul.
II-2.2. La mémoire centrale.
II-2.3. Les unités d'échange et unités de liaison.
II-2.4. Les unités de disques.

II-2.4.1. Les unités de disques à têtes fixes et les tambours.
II-2.4.2. Les unités de disques à têtes mobiles.

II-2.5. Les périphériques autres que les disques et la console-opérateur.

II-2.5.1. Les périphériques n'utilisant pas de fichiers symbionts.
II-2.5.2. Les périphériques utilisant des fichiers symbionts.

II-2.6. La console-opérateur.

II-2.7. Les ordinateurs satellites.

II-3. Le fonctionnement de l'installation sous moniteur de multiprogrammation.

II-3.1. Description générale d'un système de multiprogrammation.

II-3.1.1. La gestion de la file E des travaux candidats au lancement.

II-3.1.1.1. L'entrée dans la file E.
II-3.1.1.2. La structure de la file E.
II-3.1.1.3. La sortie de la file E.

II-3.1.2. Etude de la file V.

II-3.1.2.1. Etat opérationnel d'une tâche.
II-3.1.2.2. Etat "en attente d'entrée-sortie" d'une tâche.
II-3.1.2.3. Etat prêt d'une tâche.

II-3.1.3. La gestion des entrées-sorties.
II-3.1.4. L'imputation du travail du moniteur.

II-3.2. Les paramètres probabilistes du système.

II.3.2.1. Les paramètres de passage de la file E dans la file V.
II-3.2.2. Les paramètres d'évolution de la file V.

II-3.2.2.1. A partir de l'état opérationnel
a) passage à l'état "en attente d'entrée-sortie".
b) passage à l'état "prêt".
c) passage à la fin du travail.
II-3.2.2.2. A partir de l'état "en attente d'entrée-sortie" passage à l'état actif.
a) disques à accès rapide.
b) disques à têtes mobiles.
c) les périphériques, autres que les disques et la console opérateur n'utilisant pas de symbiont.
d) les périphériques lents utilisant des symbionts.
e) la console opérateur.

II-3.2.2.3. A partir de l'état "prêt".

II-4. Conclusion.
Récapitulation des paramètres.

INTRODUCTION

INTRODUCTION

Les systèmes d'exploitation existant actuellement ou en cours d'élaboration sont d'une complexité croissante, intégrant des éléments matériels aux performances très diverses et des éléments programmés extrêmement variés tant par les services qu'ils rendent que par les coûts qu'ils entrainent.

L'abondance des possibilités entraîne la nécessité de choix qui doivent être les plus rationnels possible.

Cette situation a suscité chez les contructeurs aussi bien que chez les utilisateurs, de nombreux travaux sur l'évaluation des performances, l'optimisation et la comparaison des systèmes.

Plusieurs approches du problème ont été utilisées séparément ou de manière complémentaire : la mise au point de techniques de mesure, l'élaboration de modèles analytiques et la simulation.

Néanmoins, à l'heure actuelle, en dehors de la simulation, très coûteuse à élaborer et à mettre en oeuvre - pour cela réservée, en fait, aux contructeurs- les utilisateurs sont très démunis.

En effet en raison de la complexité des systèmes les modèles analytiques s'intéressent essentiellement à des problèmes locaux : échanges mémoire centrale-mémoire auxiliaire, gestion des travaux, gestion des disques ...

Quant aux mesures, si des techniques existent, encore faut-il déterminer ce que l'on va mesurer et par quelle méthode.

Face à cette situation il nous a paru nécessaire d'élaborer un modèle analytique global de système d'exploitation, modèle mettant en évidence les fonctions critiques régissant le comportement du système.

Pour cela nous avons utilisé la théorie des files d'attente ce qui nous a permis d'obtenir un modèle à la fois général, souple et facile à résoudre du comportement d'une installation en exploitation partagée (multiprogrammation, temps partagé etc...).

A côté de son aspect théorique, la mise en application du modèle demande la connaissance et donc l'évaluation d'un certain nombre de paramètres définissant les caractéristiques du hardware, du software et des utilisateurs de l'installation étudiée, paramètres régissant le fonctionnement du système ainsi que les fonctions critiques. Ces données vont permettre de résoudre de manière très simple et rapide (par l'analyse numérique) le système d'équations définissant le comportement de l'installation.

Par ce modèle stochastique nous pourrons non seulement étudier les performances de l'installation, mais encore évaluer l'incidence des modifications susceptibles d'être apportées au hardware, au software ou au comportement des utilisateurs, faire des comparaisons (performances, coûts) et optimiser.

CHAPITRE I

PRÉSENTATION DES TRAVAUX EXISTANT SUR L'EVALUATION DES PERFORMANCES, L'OPTIMISATION ET LA COMPARAISON DES SYSTEMES D'EXPLOITATION

I.1. Introduction.

Avant d'aborder l'étude du modèle que nous proposons il nous a paru nécessaire de faire le point des travaux existants sur l'évaluation, l'optimisation et la comparaison des performances des systèmes d'exploitation.

Notre propos n'est pas de présenter une étude exhaustive de tout ce qui a été fait dans ce domaine, mais plutôt de montrer dans quel sens se sont développées ces recherches et ce que l'on peut en attendre. Ce chapitre doit donc être envisagé comme une introduction à notre modèle, des définitions précises étant données ultérieurement.

Plusieurs approches de ce problème de l'évaluation et de l'optimisation de système ont été utilisées :

1) La mise au point de techniques de mesure sur des systèmes réels permet une approche pratique du comportement d'une installation.
2) A l'opposé,l'élaboration de modèles analytiques permet l'étude théorique de certains aspects du système.
3) Entre ces deux pôles,la simulation est bâtie à la fois sur un modèle mathématique et des données issues du monde réel.

Avant de voir de manière précise ces trois manières de procéder,il convient de noter qu'elles ont été utilisées séparément mais aussi de manière complémentaire : ainsi des mesures établies sur une installation peuvent servir à vérifier la concordance entre les résultats d'un modèle analytique et la réalité ou encore servir d'entrées à une simulation.

I.2. Les mesures :

Les mesures établies sur un système réel permettent d'obtenir a postériori des données sur le comportement effectif de l'installation.

Dans ce domaine des travaux très divers existent. En effet de très nombreux paramètres sont utiles à connaitre :

. Ainsi les performances et fréquences d'utilisation de différentes ressources hardware et software de l'installation : unité centrale, mémoire, canaux, périphériques mais aussi compilateurs, bibliothèque, processeurs et diverses sous-parties du système.
De plus, des mesures peuvent étudier le comportement interne de chaque type de ressources : fréquence et durée de certaines opérations élémentaires ou temps moyen passé dans telle partie d'un processeur.

. Les caractéristiques des programmes utilisateurs passés sur l'installation sont aussi très précieuses. Il s'agira de leur taille, des ressources demandées, du temps de calcul etc..., complétées dans le cas de systèmes conversationnels par des données sur le comportement des utilisateurs à la console (temps de réflexion).

. D'autres mesures portent enfin sur les longueurs moyennes des diverses files d'attente qui se produisent en différents points du système : file d'attente des travaux candidats à l'entrée en mémoire centrale, ou de service du disque ou encore file d'attente pour l'impression des résultats.

On pourra donc chercher à évaluer des éléments très différents. Néanmoins les données à recueillir pour effectuer ces mesures seront le plus souvent des contenus de mémoires avec éventuellement consultation d'une horloge.

Les données recueillies sont rangées dans des fichiers sous forme élémentaire pour être étudiées ultérieurement ou encore mises, au fur et à mesure de leur collecte, dans des

histogrammes afin de réduire leur volume.

La diversité des mesures possibles entraîne une grande variété des techniques utilisées pour les obtenir. Elles se regroupent en deux grandes familles que nous allons étudier successivement : les mesures cablées et les mesures programmées.

I.2.1. Les mesures câblées :

Les mesures câblées sont effectuées à l'aide de circuits logiques qui sont implantés sur la machine.

Cette manière de procéder a le très gros avantage de ne pas perturber le fonctionnement de l'installation.

Par suite il est possible d'obtenir des informations dont la résolution est extrêmement fine (la microseconde) sur les différentes parties du système (mémoire, canaux, unité centrale etc ...).

De plus on pourra installer des points de mesure en grand nombre : ainsi l'ordinateur TX2 du Massachussetts Institute of Technology Laboratory Lincoln (Réf. Nemeth [35]) est muni d'un équipement donnant accès à cent-soixante points de l'unité centrale.

Ces techniques permettent donc de suivre de très près, éventuellement à l'aide d'un autre ordinateur dont le seul travail est d'analyser les données au fur et à mesure de leur arrivée, le comportement d'un ordinateur : elles sont donc d'un grand intérêt.

On ne peut que regretter qu'elles soient si peu répandues à l'heure actuelle.

I.2.2. Les mesures programmées :

Contrairement aux mesures câblées, les mesures programmées ont une influence sur le comportement de l'installation qu'elles étudient. En effet elles ont besoin pour s'effectuer de temps de calcul et de place en mémoire ou sur des

fichiers pour garder les résultats collectés,et dans ce dernier cas elles utiliseront donc aussi des canaux.

Ce temps d'utilisation des ressources,soustrait à son usage normal,sera d'autant plus important que les mesures seront plus nombreuses : qu'elles portent sur plusieurs points ou qu'elles se produisent très souvent dans le temps.

Deux types de mesures programmées se dégagent selon que l'on veut étudier des performances locales de l'installation, comme les performances d'une unité centrale, d'un processeur ou d'un travail utilisateur, ou au contraire les performances globales d'un ordinateur sous le contrôle d'un moniteur et avec une certaine charge de travaux.

I.2.2.1. Les mesures locales au système :

La variété des grandeurs à mesurer entraîne une grande diversité dans les techniques mises au point, nous allons en voir quelques exemples.

I.2.2.1.1. Mesures sur le matériel : les performances d'unités centrales :

Deux techniques visant à établir et à comparer les performances d'unités centrales sont particulièrement connues et utilisées : la méthode des assortiments d'instructions ("instructions mixes") et la méthode des noyaux ("kernels") (Réf. Calingaert [7] , Drummond [13], Manou [33] , Groupe d'évaluation [51]).

I.2.2.1.1.1. Les assortiments d'instructions :

Dans la méthode des assortiments d'instructions on attribue à chaque instruction ou type d'instruction du répertoire d'un calculateur un facteur de pondération, sensé représenter sa fréquence d'occurence, et on en déduit le temps moyen d'exécution d'une instruction, ou d'un type d'instruction.

Cette méthode, simple dans son principe, repose sur le problème statistique de la détermination des facteurs de pondération.

Les difficultés de mise en application qui en résultent sont illustrées par Calingaert (Réf. [7]). Celui-ci cite en effet les résultats obtenus par dix spécialistes concernant le temps moyen d'exécution des instructions de comparaison sur I.B.M. 360/40 : ces résultats s'échelonnent entre 11,88 et 30,66 avec pour moyenne 21,5 et pour écart-type 7,0.

I.2.2.1.1.2. Les noyaux :

La méthode des noyaux est un perfectionnement de la méthode des assortiments d'instructions.

Elle est basée sur les temps d'exécution non d'instructions mais de fonctions (ou noyaux) composées d'un certain nombre d'instructions. Selon les cas le nombre d'instructions sera petit, il s'agira alors de noyaux du type comparaison de deux quantités avec branchement si elles sont égales, ou encore addition de deux quantités le résultat étant rangé à un troisième emplacement, ou il pourra être grand si on prend pour noyau une inversion de matrice ou l'évaluation d'un polynôme.

On se servira d'assortiments d'instructions pour déterminer la durée d'exécution des différents noyaux.

Un programme est considéré comme un ensemble de noyaux, son temps d'exécution sera donc la somme des temps d'exécution des différents noyaux pondérée par leurs fréquences d'utilisation.

Cette méthode, basée sur le même principe que la précédente, présente des difficultés analogues pour son application,difficultés liées à la détermination des facteurs de pondération.

Néanmoins elle a pu servir,par exemple,pour prédire, à partir de programmes existants sur 7090, les performances de l'I.B.M. 360.

En conclusion à ces deux méthodes il faut dire que, ne comportant pas d'opérations d'entrée-sortie, elles ne peuvent être utilisées pour évaluer des sytèmes et sont donc limitées à l'étude des performances d'unités centrales.

I.2.2.1.2. Mesures de programmes :

Des mesures de programmes sont souvent nécessaires, pour étudier certaines caractéristiques des travaux utilisateurs traités sur l'installation ou pour suivre le comportement de modules du moniteur dans différents cas afin de les optimiser.

Deux méthodes sont là aussi particulièrement intéressantes : l'une d'elles utilise les déroutements, et l'autre est la trace.

I.2.2.1.2.1. Méthode par déroutements :

Cette méthode consiste à laisser le programme à mesurer se dérouler normalement jusqu'à l'arrivée dans le programme, d'un événement servant à la mesure. Il se produit alors un déroutement, les opérations nécessaires à la mesure sont effectuées, puis le contrôle est rendu au programme interrompu qui va pouvoir continuer à se dérouler jusqu'à l'arrivée du déroutement suivant.

Pour préciser donnons un exemple de cette méthode :

Supposons que l'on veuille étudier avec quelle fréquence arrivent dans un programme particulier des instructions ayant certaines caractéristiques, par exemple des instructions de rupture de séquence ou bien d'appel - moniteur d'un type bien précis.

Pour cela le module à tester va être, avant son exécution, pris en compte par un module qui va examiner successivement toutes ses instructions et par exemple remplacer le code-opération des instructions intéressantes par un code d'instruction inexistante.

L'exécution du programme transformé sera ensuite lancée. Chaque fois qu'une instruction modifiée sera décodée il se produira un déroutement pour instruction inexistante.

Ce déroutement, au lieu d'être traité par le moniteur de la manière habituelle, sera récupéré par un module

de mesure joint au système. Ce module prendra alors le contrôle, mettra à jour ses tables, rétablira l'instruction piégée dans sa forme initiale puis rendra le contrôle au programme interrompu au niveau de cette instruction.

I.2.2.1.2.2. La trace :

La trace est une méthode d'évaluation de programmes très souvent utilisée.

En effet elle permet de suivre, instruction par instruction, le déroulement de tout ou partie d'un processus et par suite d'avoir des renseignements aussi complets que l'on veut sur son comportement.

La contrepartie de cette abondance d'informations est la lenteur : plus on voudra extraire de renseignements, plus la trace sera coûteuse.

Cette méthode est utilisée, entre autres, pour suivre l'évolution d'arbres de recouvrement (Réf. Spacek [46]), pour voir la répartition des demandes de pages (Réf. J.L. Smith [45]), pour étudier un échantillon de programmes utilisateurs (Réf. Cheng P.S. [8]) etc ...

I.2.2.1.3. Etude de comportement des utilisateurs :

Le comportement des utilisateurs intervient de manière indirecte dans le fonctionnement de l'installation par l'intermédiaire des travaux qui sont traités.

Mais lorsque le système est conversationnel les utilisateurs ont de plus une intervention directe.

C'est pourquoi des études se sont consacrées à l'évaluation du "temps de réfléxion" de l'utilisateur à sa console : c'est le temps écoulé entre la fin d'un message envoyé par la machine sur sa console et la réponse donnée par l'utilisateur.

J. Maublanc (Réf. [62]) s'est intéressée, pour sa part, à l'étude statistique, sur le système en temps partagé

I.B.M. 360 RAX, des instants d'arrivée des utilisateurs dans le système (étude,pour une journée,des temps entre deux arrivées successives, répartition des appels au cours de la journée) et à l'étude de la durée des communications.

I.2.2.2. Les mesures globales :

Les mesures locales ne s'intéressent qu'à un aspect du système. Cette manière de faire peut donner des renseignements très précieux sur le fonctionnement de l'installation.

Elle est malgré tout très incomplète pour l'évaluation du fonctionnement du système dans son ensemble, sauf dans le cas où diverses mesures locales sont utilisées comme entrées d'un modèle analytique ou de simulation étudiant le comportement global de l'installation.

A l'exception de ce cas les mesures globales sont donc nécessaires.

Ces mesures sont faites soit lors du passage de jeux d'essais de programmes ("benchmarks" cf. Groupe d'évaluation RIRO Réf.[51]) qui constituent un échantillon représentatif d'un certain type de charge, soit au cours du fonctionnement normal de l'installation.

Un but n'est pas de tout mesurer: ce serait à la fois très onéreux et inutile car le temps consacré aux mesures viendrait en fausser de manière importante les résultats. Il s'agit au contraire d'établir un certain nombre de points de mesure aux endroits les plus significatifs du système.

Nous citerons deux types de mesures globales : les mesures de type "comptabilité" et les mesures par échantillonnage.

I.2.2.2.1. Les mesures de type "comptabilité" :

Le module de comptabilité d'un moniteur est mis en oeuvre à l'arrivée d'un certain nombre d'événements (début ou fin d'intervalle d'exécution d'un travail, fin d'entrée/sortie, mise en oeuvre d'une ressource etc ....).

Les événements recueillis sont répertoriés dans des tables et seront utilisés ultérieurement pour faire la comptabilité par travail mais aussi pour extraire des statistiques sur le système.

Cette manière de procéder est très intéressante mais son implantation sur les machines est faite dans un but comptable et guère en vue d'une évaluation autre que très grossière du comportement de l'installation : en effet le nombre des rubriques auxquelles on a accés est relativement réduit et les résultats ne peuvent être analysés faute de pouvoir être décontractés.

De plus il n'est pas toujours aisé de savoir de manière très précise ce qui est comptabilisé sous les différentes rubriques.

L'utilisation de la comptabilité dans un but d'évaluation a été réalisée par Reinier(Réf.[42]). Ce procédé permet d'avoir des informations sur tous les travaux passés sur l'installation, et ceci à un coût minimum puisque la comptabilité est faite de toute façon. Mais les renseignements fournis sont insuffisants pour une étude approfondie.

En fait, il semble que des informations beaucoup plus intéressantes pourraient être obtenues si on utilisait non pas les résultats de la comptabilité mais le processus de collecte des données qui lui sont nécessaires. Ces données pourraient être rassemblées en fichiers pour une exploitation ultérieure et non simplement agrégées de la manière que nous avons dite. Il faudrait alors envisager d'augmenter le nombre des rubriques afin de pouvoir utiliser vraiment ces mesures pour une étude approfondie.

I.2.2.2.2. Les mesures par échantillonnage :

A l'opposé de la comptabilité qui est mise en oeuvre quand surviennent certains événements dont on pourra aller chercher l'heure d'arrivée, la technique de mesure par échantillonnage est mise en oeuvre par un signal d'horloge qui crée

une interruption : le module de mesure va alors chercher le contenu d'un certain nombre de mots de mémoire, après quoi la situation antérieure est rétablie et le contrôle revient au processus interrompu.

Selon les mots qu'on aura choisi de consulter, on pourra obtenir la longueur de différentes files d'attente, le type de l'instruction en cours au moment de l'interruption, son adresse, le nombre de canaux actifs etc... Autant de renseignements qui permettront d'établir le comportement du système, en moyenne, et pourront mettre en évidence certaines de ses lacunes.

Selon la priorité donnée au module d'échantillonnage il s'appliquera à tout le système ou simplement à certains programmes déterminés.

La fréquence des interruptions, qui pourront se faire à l'issue d'intervalles de temps de durée déterminée ou aléatoire, sera, là aussi, un compromis entre la précision et le coût des mesures. Néanmoins il semble qu'en général la perte de temps soit de 3 à 5% ,ce qui est peu de chose (Réf.M.de Saint Seine [52] , E.N.I.C.A.[55], M.F. Pistré[69]).

Il semble qu'il faille délimiter le rôle des techniques d'échantillonnage programmées à un domaine précis concernant des taux d'utilisation d'éléments matériels (Ex. canaux, types d'instruction etc...) ou programmés (Ex. tel processeur ou telle partie du moniteur ou d'un programme etc ...) ou encore à des longueurs moyennes de files d'attente.

Mais il semble illusoire de vouloir leur demander des résultats sur des fréquences d'arrivée d'événements dans le temps (Ex. fréquence d'arrivée de demandes d'entrées-sorties pour les utilisateurs, ou fréquence en multiprogrammation du passage du contrôle de l'unité centrale d'un utilisateur à un autre).

En effet, s'il est vrai qu'à la limite la méthode d'échantillonnage donne pratiquement tous les renseignements que l'on veut sur le système, mais à quel prix (perturbation et coût), puisqu'il suffit pour cela d'avoir une fréquence

d'échantillonnage très fine, il est non moins vrai que de tels renseignements peuvent être donnés, pour une perturbation et un coût minima, par la méthode exposée précédemment (de type "comptabilité" cf.I.2.2.2.1. dernier paragraphe).

Pour conclure cette partie sur les mesures nous devons dire que si certaines, comme les mesures établies par échantillonnage ou les évaluations de programmes sont exploitables immédiatement pour une optimisation, d'autres au contraire ne prennent tout leur intérêt que lorsqu'elles sont utilisées pour l'application de modèles analytiques ou la mise en oeuvre de simulation.

I.3. Les modèles analytiques :

De nombreux modèles analytiques ont été élaborés pour étudier le comportement des systèmes informatiques.

Ils essaient de mettre en évidence les structures mathématiques sous-jacentes afin d'établir des relations entre différents éléments du système analysé.

Il ne saurait être question pour de tels modèles de refléter toute la complexité des systèmes étudiés mais au contraire de mettre en lumière leur fonctionnement, souvent sur des points très précis, à l'aide d'un petit nombre de paramètres bien choisis.

Les paramètres utilisés sont rarement déterministes (exception : le modèle déterministe proposé par Rousset de Pina Réf. [43]) ,puisqu'on modélise des phénomènes de nature aléatoire. Par suite on fera souvent des hypothèses, qui peuvent être suggérées par des mesures, sur leur distribution.

Pour être utilisable un modèle doit être le plus proche possible de la réalité. Ceci nécessite une certaine souplesse et est difficilement compatible avec une simplicité suffisante pour le rendre soluble, au moins numériquement.

Malgré leurs limites les modèles proposés sont nombreux car, en dehors de leur intérêt théorique, ils permet-

tent, selon le cas, de faire des évaluations, des comparaisons, des optimisations, de tester des hypothèses le plus souvent sur des points très précis, et cela a priori c'est-à-dire sans nécessiter l'existence physique du système étudié.

Comme dans le cas des mesures, nous trouvons parmi les modèles deux types que nous étudierons successivement : les modèles locaux, c'est-à-dire étudiant un point particulier du système, et les modèles plus globaux, qui essaient d'appréhender le système, sinon dans son ensemble, du moins de manière beaucoup plus large.

I.3.1. Les modèles locaux

Les modèles locaux sont de loin les plus nombreux.

Ceci s'explique par la nécessité d'études approfondies des points particulièrement délicats, et aussi par la difficulté, nous le verrons, d'établir des modèles globaux.

Les nombreuses publications sur les modèles locaux peuvent se répartir, pour la plupart, en trois groupes :

. les modèles de gestion de l'ensemble des travaux (scheduling).
. les modèles de gestion de la mémoire centrale.
. les modèles de gestion de disques.

I.3.1.1. Les modèles de gestion de l'ensemble des travaux.

Ce sont les plus abondants.
(Réf : Mc KINNEY [31] , E.G. COFFMAN JR [9] , BERNSTEIN [3] , WAH-CHUN CHAN [49] , HEACOX [19] , LIU [28] , IGAL ADIRI [1], etc)

Il s'agit essentiellement de modèles traitant d'installations en temps partagé.

Ils étudient le phénomène suivant : les tâches entrant dans le système sont placées en files d'attente, selon l'algorithme de gestion du régulateur, en attendant d'être servies.

Lorsque son tour est arrivé, chaque tâche a le contrôle de l'unité centrale pendant une période de temps appelée quantum.

Si durant le quantum la tâche est terminée elle sort et le service de la tâche suivante commence, sinon la tâche rejoint une file où elle attendra de prendre à nouveau le contrôle de l'unité centrale.

Le temps passé par la machine en gestion de l'ensemble des travaux, allocation, gestion des mémoires-tampons et contrôle des entrées-sorties des consoles est inclus dans le temps de "supervision" du moniteur.

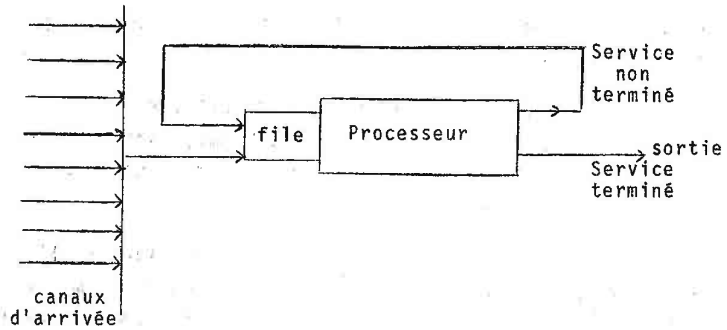
Si aucune des tâches ne peut prendre le contrôle, le temps d'unité centrale pendant lequel on entre en mémoire une nouvelle tâche à activer est perdu.

Dans les travaux étudiant ce problème on trouve de très nombreux développements sur les avantages respectifs des modèles de gestion des travaux: modèle tourniquet ("round-robin "), modèles à 2 ou n niveaux de priorités internes, modèles à priorités externes, la priorité pouvant être statique ou dynamique. (cf figure p 16).

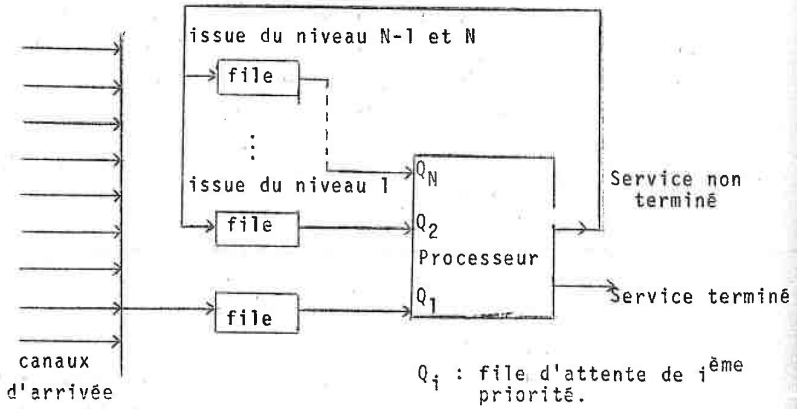
En général ces modèles diffèrent seulement par le choix des paramètres suivants :

. nombre de canaux d'entrée fini ou infini,
. nombre d'unités centrales,
. types de processus d'arrivée,
. loi du temps de calcul,
. prise en compte ou non du temps d'échange mémoire centrale - mémoire secondaire (swapping) et du temps de supervision, si oui leur somme pouvant être constante ou aléatoire,
. hypothèses sur la durée du quantum de temps.

Ces modèles donnent lieu à de très nombreux calculs théoriques basés essentiellement sur les chaînes de Markov, les espérances conditionnelles et la théorie des files d'attente



Modèle R.R. (round-robin) ou "tourniquet"



Q_i : file d'attente de $i^{ème}$ priorité.

Modèle avec N niveaux de priorités internes.

Les résultats attendus concernent le temps moyen de réponse, le temps moyen d'attente dans la file (éventuellement en fonction du temps de service requis et/ou du niveau de priorité) et la longueur moyenne de la file (éventuellement pour chaque niveau de priorité).

I.3.1.2. Les modèles de gestion de la mémoire.

L'extension de la multiprogrammation et de l'usage de la mémoire virtuelle a posé avec acuité le problème de l'optimisation de la gestion de la mémoire centrale.

Les modèles de gestion de mémoire contribuent à l'étude de ce problème sous différents aspects et dans le cas de mémoires paginées le plus souvent.

Ainsi Denning (Réf [11]) met en évidence l'existence et les proprietés de "l'espace de travail" (working-set) qui pour un programme est le plus petit ensemble de ses pages devant résider en mémoire centrale pour que le programme s'éxécute avec un degré désiré d'efficacité. Il en déduit les causes de l'échec des algorithmes de remplacement des pages en mémoire (sélection des pages à sortir au hasard ou sortie de la page qui n'a pas été utilisée depuis le plus longtemps, ou première-entrée - première-sortie etc ....).

Cette notion d'espace de travail prend tout son intérêt dans les systèmes de multiprogrammation faisant entrer en mémoire les pages à la demande. Ce problème est étudié à l'aide d'une chaine de Markov par Wallace (Réf.[50]) qui se penche sur les relations entre le nombre de programmes auxquels on permet d'être simultanément présents en mémoire, les taux de trafic du tambour et l'utilisation de l'unité centrale.

Quant à Oden (Réf.[38]), il étudie le comportement du contenu de la mémoire lorsqu'on utilise la pagination.

L'étude se fait à l'aide de processus semi-markoviens et les résultats obtenus relient, sous certaines hypothèses, le temps moyen d'exécution d'une charge multiprogrammée, le

nombre de pages de mémoire allouées à un programme qui s'exécute et la fraction de temps d'exécution d'un programme avec la variabilité statistique de la charge en programmes dans un environnement de page-à-la-demande avec des partitionnements dont la taille peut varier dans le temps.

Enfin les problèmes de partitionnement statique et dynamique de la mémoire en multiprogrammation sont étudiés par Coffman et Thomas (Réf.[10]) qui utilisent un processus gaussien stationnaire pour modeliser la demande d'espace des programmes. Ils justifient l'emploi de ce processus en considérant que :

. le fait que la distribution soit définie pour des valeurs aussi bien positives que négatives de la taille de l'espace de travail (working set) n'est guère ennuyeux à cause des valeurs des paramètres m (moyenne) et σ (écart-type).

. le fait d'utiliser une distribution normale, alors que les tailles des espaces de travail prennent seulement des valeurs discrètes, n'entraîne qu'une faible erreur lorsqu'on s'intéresse surtout aux applications dans lesquelles la taille de l'espace de travail varie dans un nombre suffisamment grand de valeurs.

I.3.1.3. Les modèles de gestion des disques :

Les disques jouent un rôle considérable en multiprogrammation : ils font en effet de très nombreux échanges avec la mémoire centrale et peuvent pénaliser lourdement l'ensemble de l'installation si leur gestion est mal conçue.

Un certain nombre de publications existent sur ce problème.

Nous citerons Burge (Réf.[5]) qui étudie, à l'aide de chaînes de Markov, deux modes d'accès, sur un ensemble formé d'un disque à tête fixe et d'une mémoire-tampon (buffer). Il en déduit sous certaines hypothèses, le nombre moyen de demandes satisfaites en un tour complet du disque.

Teorey (Réf.[48]), quant à lui, compare cinq techniques de gestion des demandes d'entrées-sorties de disque à tête mobile :

. Premier entré premier servi.

. La demande à laquelle on peut accéder le plus rapidement est servie la première.

. La méthode de scrutation (scan) : le bras de l'unité de disque se comporte comme une navette : il balaie successivement tous les cylindres, de l'intérieur vers l'extérieur, puis vice versa, en satisfaisant les demandes au fur et à mesure.

. Si l'on accepte que la tête de lecture puisse faire au maximum N tours (au lieu d'un) sur le même cylindre la méthode sera dite "de scrutation d'ordre N".

. Le schéma d'Eschenbach : il est dit d'ordre E si la tête de lecture fait au moins E tours par cylindre, les cylindres étant parcourus dans l'ordre 0, 1....N puis à nouveau 0, 1 ...

Il utilise comme critère de comparaison des différentes méthodes le temps de recherche escompté et le temps d'attente escompté.

Avant de terminer cette partie sur les modèles locaux nous devons préciser que tous les modèles locaux n'entrent pas, bien sûr, dans les trois groupes précités. Comme il n'est pas question de faire une étude exhaustive de tous les modèles existants nous nous tiendrons donc à ceux là qui nous semblent les plus représentatifs des travaux faits sur les modèles de systèmes.

I.3.2. Modèles globaux .

A côté des modèles étudiés précédemment et qui s'intéressent à des points bien précis de l'installation quelques modèles plus globaux ont été établis.

Ainsi Alan SCHERR a construit, à l'aide de la théorie des files d'attente, un modèle servant à l'étude du temps de réponse du système conversationnel CTSS en fonction du nombre de consoles. Il suppose que les demandes provenant des consoles utilisateurs sont traitées dans leur ordre d'arrivée et l'une à la suite de l'autre (il n'y a donc pas de multiprogrammation).

Lors d'une session deux états sont possibles, et se succèdent, pour un utilisateur : réflexion avant de lancer une nouvelle demande à la console, ou attente de réponse à une demande.

Ces hypothèses permettent à Scherr d'établir un modèle à la fois simple et efficace.

James D. Foley (Réf.[15]) a étudié le Système Exécutif de l'Université de Michigan (U.M.E.S.) implanté sur I.B.M. 7090. Son modèle est basé sur les neuf états suivants :

1) Entrée pour trouver le début d'un nouveau travail.

2) Examen des cartes commandes pour déterminer si des traductions sont demandées.

3) Examen des cartes commandes pour déterminer quel traducteur est requis.

4) Compilation MAD.

5) Assemblage UMAP.

6) Compilation Fortran.

7) Travail chargé en mémoire pour exécution.

8) Exécution du travail.

9) Production d'une image mémoire.

Foley établit les probabilités de transition entre ces neuf états à partir de statistiques établies sur I.B.M. 7090.

La conclusion qu'il tire est qu'à partir d'un modèle simple il arrive à des résultats intuitivement corrects et qui peuvent être un indice utile des performances du système.

Smith enfin (Réf.[44]) a utilisé un processus de Markov continu pour la représentation d'un système en temps partagé utilisant la pagination.

En conclusion sur les modèles analytiques nous pouvons dire que si de nombreux travaux se sont intéressés à des parties locales des systèmes d'exploitation, seules quelques études ont essayé une approche plus globale.

Ces essais sont néanmoins limités dans leur application. Plusieurs raisons contribuent à cette situation. Tout d'abord la réalité à étudier est ici extrêmement souple, variée et complexe : on aura donc des difficultés à la traduire dans des modèles mathématiques pouvant être résolus et exploités.

Par suite il semble que l'on se soit tourné vers la simulation pour étudier le comportement global d'installations. Les modèles locaux peuvent d'ailleurs, en dehors des résultats théoriques qu'ils apportent, être intégrés dans certaines parties d'une simulation pour alléger cette dernière et la rendre plus efficace.

Aussi allons-nous nous pencher maintenant sur les modèles de simulation qui ont été établis.

I.4. La simulation des systèmes :

La simulation, nous l'avons dit, est à la jonction des approches du comportement des systèmes par les mesures et par les modèles analytiques.

En effet elle a besoin pour son élaboration d'intégrer dans un modèle mathématique des données du monde réel. De plus la validité d'une simulation doit être testée : pour cela, si on ne peut pas comparer les résultats obtenus avec des résultats réels, au moins pour des cas particuliers ou des situations

voisines, on ne pourra alors que vérifier intuitivement la vraisemblance des résultats.

L'avantage de cette technique est de permettre, si le modèle est valide, d'approcher la réalité d'aussi près que l'on veut.

Mais en contrepartie, plus le modèle colle à la réalité, surtout en ce qui concerne les systèmes d'exploitation, plus il devient lourd et complexe à élaborer et à mettre en oeuvre (encombrement, temps de calcul). A titre d'exemple Norman Nielsen (Réf.[36],[37]) indique que la simulation d'une minute de temps partagé sur B.6500 demande 5 à 10 minutes de B.5500 ou encore qu'il faut une minute d'I.B.M. 360/50 pour simuler 2 ou 3 minutes d'I.B.M. 360/67.

Ce coût élevé explique pourquoi la simulation a été souvent mise en oeuvre par des constructeurs : elle sert alors à la conception et à l'amélioration d'un système ou des parties d'un système.

Enfin il faut noter que si les modèle analytiques ne sont pas liés strictement à telle ou telle installation, ceci, par nature, n'est pas vrai pour les simulations qui se rapportent de manière précise aux caractéristiques propres de l'installation simulée. Ceci est d'autant plus vrai que la simulation se rapproche plus de la réalité.

Nous n'entrerons pas dans les détails de ces travaux. En effet, en dehors de la méthodologie de la simulation, on y trouve essentiellement des descriptions très précises du système à simuler.

Et nous citerons simplement quelques travaux de simulation, à titre d'exemples :

. la comparaison d'algorithmes de gestion de disques par Teorey et Pinkerton (Réf.[48]). Ces auteurs ont utilisé l'approche par un modèle analytique et par la simulation.

. L'étude d'algorithmes d'ordonnancement pour le Time-Sharing Operating System du R.C.A. par Oppenheimer et Weizer (Réf.[54]).

. La simulation d'un système conversationnel CASSCOU par Joëlle Coutaz (Réf.[61]).

. La simulation de systèmes en temps partagé par Nielsen (Réf.[36]) pour I.B.M. 360/67.

. Et, de Nielsen aussi, la simulation de différentes techniques utilisées en time-sharing : la relocation de programmes, la minimisation des délais dûs à la rotation des disques et la minimisation du volume des échanges dûs au swapping pour B.6500 (Réf.[37]).

I.5. Conclusion :

Pour conclure ce chapitre consacré aux différents types de travaux effectués sur les problèmes d'évaluation, d'optimisation et de comparaison de systèmes, force est de constater leur grande dispersion, et l'on voit apparaître tout l'intérêt de la construction d'un modèle global, intégrant les résultats de mesures bien choisies et donnant, de manière beaucoup plus simple et rapide que la simulation une image du comportement d'une installation.

CHAPITRE II

DESCRIPTION D'UN SYSTEME DE MULTIPROGRAMMATION

II.1. Introduction :

Le travail que nous présentons était basé, à l'origine, sur l'étude de l'ordinateur C.I.I. 10.070.

En vue d'évaluer et d'optimiser les performances de l'installation, nous avons essayé d'établir un modèle stochastique de son comportement sous le moniteur de multiprogrammation SIRIS 7.

Nous avons constaté alors que la description du comportement du système d'exploitation par un réseau de files d'attente permet d'obtenir un ensemble d'équations aisément soluble.

Ceci est dû, nous le verrons, aux relations liant ces équations les unes aux autres.

Ce type de relations n'est, bien sûr, pas spécifique à SIRIS 7, aussi avons-nous pû, dans une seconde phase de notre travail, appliquer le même raisonnement et la même méthode à des systèmes d'exploitation différents mais ayant en commun cette caractéristique que nous avions mise en évidence.

Aussi dans la suite de ce chapitre allons-nous décrire un système de multiprogrammation en vue de faire apparaître les paramètres mathématiques qui seront nécessaires à l'établissement de notre modèle.

Ce modèle, on le verra, est suffisamment souple pour pouvoir être adapté aux caractéristiques propres à une installation. Il ne sera donc pas question d'entrer dans des particularités locales qui auraient leur place dans des applications particulières du modèle et non ici.

C'est donc dans cette optique probabiliste que nous allons étudier successivement les caractéristiques du matériel qui nous semblent essentielles du point de vue du comportement du système, puis les caractéristiques du

comportement proprement dit d'un système de multiprogrammation.

II.2. Description du matériel :

La configuration de base est composée d'une mémoire centrale en liaison directe avec une (ou plusieurs) unité centrale et en liaison par l'intermédiaire d'unités d'échanges avec une (ou plusieurs) unité de disques rapides, une (ou plusieurs) unité de disques lents, d'autres périphériques rapides ou lents, y compris éventuellement des consoles des utilisateurs de temps partagé, la console des opérateurs et des ordinateurs satellites.

Avant d'approfondir l'étude de ces différents éléments nous devons établir une distinction entre les organes selon qu'ils sont à un ou plusieurs points d'accès, le nombre de points d'accès indiquant le nombre de processus pouvant utiliser l'organe en même temps.

II.2.1. L'unité centrale ou unité de calcul :

L'unité centrale exécute les instructions.

La durée d'exécution d'une instruction est variable en fonction du type (code-opération) de l'instruction, de la durée du cycle-mémoire, de l'utilisation éventuelle d'indexation et d'indirection, et aussi de la mise en oeuvre de dispositifs d'élaboration d'adresses physiques (mémoire topographique, registres de base, etc...). La durée d'exécution d'une séquence d'instructions-machine fera intervenir de plus l'utilisation éventuelle de dispositifs hardware comme le recouvrement (utilisation du temps d'écriture en mémoire pour effectuer simultanément une lecture dans un autre bloc de la mémoire), l'exécution parallèle d'instructions-machine etc...

Pour évaluer la durée moyenne d'exécution d'une instruction, τ, il faut aussi faire intervenir la fréquence d'utilisation des différents types d'instructions. Ceci dépend du genre de programmes traités sur l'installation : ainsi

Cobol, Snobol ou Fortran n'auront pas du tout les mêmes taux d'utilisation des différentes instructions (se reporter pour le problème de l'évaluation de τ à I.2.2.1.1.).

Il pourra être intéressant de choisir τ comme unité de temps, si τ est connu avec précision. Sinon on pourra prendre comme unité la durée du cycle de la mémoire.

II.2.2. La mémoire centrale :

La mémoire centrale est composée de cases pouvant contenir des mots.

L'allocation de l'espace mémoire se fait par groupes de cases de taille fixe (pages), on parle d'allocation par pages, ou de taille variable, il s'agira alors d'allocation par zones.

L'unité d'allocation est donc soit la page, soit la "case". On appellera L le nombre d'unités d'allocation en mémoire centrale.

Des mécanismes, comme la topographie mémoire lorsqu'on utilise la pagination, ou les registres de base dans l'allocation par zones, permettent la conversion entre les adresses émises par les instructions (adresses virtuelles) et les adresses réelles. Par suite, du point de vue de l'allocation, la situation physique des différentes zones ou des différentes pages n'interviendra pas.

II.2.3. Unités d'échange et unités de liaison :

Le système d'entrées-sorties est composé des périphériques et des organes permettant de mettre les périphériques en relation avec la mémoire centrale : ce sont les unités d'échange et les unités de liaison.

L'utilisation d'unités d'échange permet d'assurer le fonctionnement simultané de l'unité centrale et des périphériques.

Les unités d'échange ont accès à la mémoire. Elles peuvent gérer un ou plusieurs périphériques, selon leur type (unité d'échange simple ou multiple).

Des unités de liaison servent d'intermédiaires entre une unité d'échange et les périphériques qu'elle gère, une unité de liaison pouvant elle-même supporter plusieurs périphériques.

Chaque unité de liaison est associée à un sous-canal de l'unité d'échange.

Par suite, des périphériques situés sur des unités de liaison différentes peuvent travailler simultanément, il n'en résulte pas de retard.

En revanche un retard pourra se produire si plusieurs périphériques sont situés sur la même unité de liaison. Dans ce cas il ne peut se produire à chaque instant que des transferts d'information concernant un seul périphérique, les autres périphériques situés sur la même unité de liaison pouvant utiliser ce temps pour des positionnements.

Par suite lorsqu'on a plusieurs périphériques placés ainsi sur une unité de liaison, on peut considérer, et c'est ce que nous ferons, que cela revient, pour chacun, à voir son temps de réponse moyen allongé, et il faudra faire intervenir cet allongement dans les caractéristiques probabilistes. Ou bien on peut considérer que par exemple deux unités de disques à têtes mobiles placés sur une même unité de liaison sont équivalentes à une unité de disque de volume double et à accès plus rapide.

II.2.4. Les unités de disques :

Nous trouvons deux types d'unités de disque que nous

allons étudier successivement : les premières utilisent des têtes de lecture-écriture fixes alors que les autres utilisent des têtes mobiles.

II.2.4.1. Les unités de disques à têtes fixes et les tambours :

Pour cette étude nous pouvons considérer que le fonctionnement des tambours est analogue à celui des disques à têtes fixes.

L'utilisation de têtes fixes de lecture-écriture élimine les mouvements de bras pour avoir accès à l'information. Ces disques sont donc très rapides : ceci justifie leur utilisation comme mémoire auxiliaire à la mémoire centrale. On les appellera souvent disques rapides par opposition aux disques à têtes mobiles ou disques lents.

Les informations sont rangées sur disques rapides en blocs physiques de taille fixe, les pages, analogues aux pages de la mémoire centrale lorsqu'on utilise la pagination de la mémoire, et réparties en k_1 secteurs.

Un tour de disque se fait en θ_1 unités de temps : il faudra donc θ_1/k_1 unités de temps pour lire ou écrire une page d'information sur disque.

La répartition des demandes de lecture ou d'écriture de pages selon les différents secteurs est supposée uniforme, c'est-à-dire que lorsqu'une demande de page arrive elle a la même chance $1/k_1$ d'être destinée à chacun des k_1 secteurs.

L'utilisation des disques comme mémoire auxiliaire à la mémoire centrale entraîne que les échanges de pages entre disques et mémoire peuvent être très nombreux. Il peut alors arriver qu'une demande de lecture ou d'écriture d'une page sur un secteur survienne alors que la demande précédente sur le même secteur n'a pas encore été satisfaite. Ceci va nécessiter l'introduction de files d'attente de service du disque, au nombre de k_1, une par secteur. Nous les appellerons Q^i, $i = 1 \ldots k_1$.

Il est à noter que nous faisons ici l'hypothèse suivante : s'il existe une demande concernant un secteur lorsque le début de celui-ci passe sous les têtes de lecture-écriture, cette demande sera servie. Ceci justifie l'existence de nos k_1 files.

Toutefois le service des demandes peut se faire dans un ordre différent, par exemple l'ordre d'arrivée des demandes. Dans ce cas il faudra utiliser une seule file Q ; l'étude de son comportement sera analogue à celle de la file D des demandes de service aux disques lents, que nous verrons ultérieurement.

Nous supposerons dans un premier temps que la gestion des disques rapides est faite de manière à éviter la saturation des disques, c'est-à-dire une situation dans laquelle une demande d'écriture d'une page dans un secteur ne pourrait être satisfaite par suite de l'absence de pages libres dans ce secteur.

Nous reviendrons ultérieurement sur les problèmes de saturation (cf II.3.2.1.).

II.2.4.2. Les unités de disques à têtes mobiles :

Pour les disques à têtes mobiles, les mouvements de bras occasionnent des pertes de temps : ceci justifie le développement de différentes politiques de gestion de ces disques visant à une optimisation du temps de réponse (cf I.3.1.3.). Nous entendons par temps de réponse le temps écoulé entre le lancement d'une demande d'information et le moment où cette information est disponible à l'endroit désiré. Il est la somme du temps d'accès à l'information et de son temps de transfert.

De même que les disques à têtes fixes les disques à têtes mobiles sont divisées en secteurs, au nombre de k_2, et font un tour en θ_2 unités de temps.

Plusieurs demandes de service peuvent être présentes simultanément. Ces demandes, placées dans une file d'attente D,

seront servies en fonction de l'algorithme de gestion des entrées-sorties du disque.

Mais, si un disque à têtes fixes peut servir, en un tour, une demande par secteur, soit k_1 demandes, en revanche un disque à têtes mobiles ne pourra en servir qu'en moyenne $k'_2 \leq k$ lorsqu'il y a des demandes en attente de service. Pour certains algorithmes de gestion des demandes de service il est possible que k'_2 soit fonction de d, la longueur de la file D (cf I.3.1.3.).

II.2.5. Les périphériques autres que les disques et la console opérateur :

Les périphériques autres que les disques et la console-opérateur peuvent se classer en deux catégories selon que les informations qu'ils ont à entrer ou à sortir de la mémoire centrale transitent ou non par des fichiers intermédiaires situés sur mémoire secondaire (disques ou bandes magnétiques) et que nous appellerons fichiers-symbionts.

II.2.5.1. Les périphériques n'utilisant pas de fichiers-symbionts :

Dans cette catégorie nous trouvons les bandes magnétiques, les consoles-utilisateurs, et, selon l'installation, des périphériques lents comme par exemple lecteur ou perforateur de ruban.

Chacune de ces unités est une ressource à un seul point d'accès. Par suite seul le travail auquel est alloué la ressource peut y avoir accès et ceci pendant toute la durée de l'allocation.

Pour l'utilisateur allocataire de la ressource, une nouvelle demande de service n'est lancée que lorsque la demande précédente a été servie. La longueur d'une file d'attente étant le nombre de demandes en attente ou en cours de service, la file d'attente B^{j_s} de service au périphérique numéro j_s sera donc de longueur 0 ou 1, ce dernier cas se

présentant lorsqu'une demande est en cours d'exécution.

Etudions pour un organe le temps de réponse qui est, rappelons-le, la somme du temps d'accès et du temps de transfert.

Le temps d'accès, pour un organe, sera le plus souvent variable. Ainsi lors d'une demande d'entrée-sortie sur bande magnétique le temps d'accès à la zone de lecture-écriture dépend de l'organisation du fichier traité ("consécutif" ou "à clé") et du type d'accès utilisé (séquentiel ou direct). Lorsqu'on travaille sur console le temps d'accès à l'information est le temps, éminemment variable, de réflexion de l'utilisateur.

D'autre part pour un organe, le temps de transfert varie selon la taille des blocs physiques d'information : cette taille peut être variable ou fixée. Si elle est fixée, la taille des enregistrements logiques transférés peut, elle, être variable et ne pas coïncider avec celle des blocs physiques : ainsi le transfert d'un enregistrement pourra-t'il intéresser plusieurs blocs physiques consécutifs. De plus le temps de transfert pourra être notablement influencé par la charge de l'unité de liaison (cf II.2.3.) au moment où on veut faire le transfert.

Le temps de réponse, pour un organe, est donc variable. Lorsqu'une demande de service à l'organe n°j_s est lancée (c'est-à-dire que la file B^{j_s} est non vide) nous appellerons ζ_s dt la probabilité pour que la réponse arrive entre les instants t et t + dt.

II.2.5.2. Les périphériques utilisant des fichiers-symbionts :

Les périphériques utilisant des fichiers-symbionts sont des périphériques lents (imprimante, lecteur de cartes, etc...).

L'utilisation de ces fichiers intermédiaires, situés sur périphériques rapides (disques ou bandes magnétiques),

a pour but de diminuer la pénalisation qu'imposerait à l'ensemble de l'installation de trop fortes différences de performances entre organes.

Ces périphériques lents sont des organes à un seul point d'accès, mais leur usage différé donne aux utilisateurs l'illusion de se servir des périphériques à plusieurs points d'accès. Ainsi, si plusieurs utilisateurs, simultanément présents dans le système, demandent accès à l'imprimante, on leur accordera, en fait, à chacun, l'accès à un fichier (imprimante virtuelle) qui sera ultérieurement listé sur l'imprimante (réelle).

Nous avons supposé, dans notre modèle, que les fichiers-symbionts étaient situés sur disques rapides. On aurait pû supposer tout aussi bien qu'ils étaient sur bandes magnétiques ou sur disques lents, ceci n'entraîne alors que des modifications minimes dans l'écriture du modèle.

Nous noterons γdt la probabilité pour qu'entre les instants t et t+dt, le programme de gestion d'un périphérique lent demande l'accès à une page sur disque rapide. Dans la pratique on associera un paramètre γ_i à chaque périphérique utilisant un symbiont. On en déduira $\gamma = \sum_i \gamma_i$.

II.2.6. La console-opérateur :

La console-opérateur a accès à la mémoire centrale par une unité d'échange.

Elle travaille en prenant du temps de calcul qui est imputé, selon les cas, soit à l'utilisateur qui a nécessité l'intervention du pupitreur, soit au moniteur, comme nous l'étudierons ultérieurement.

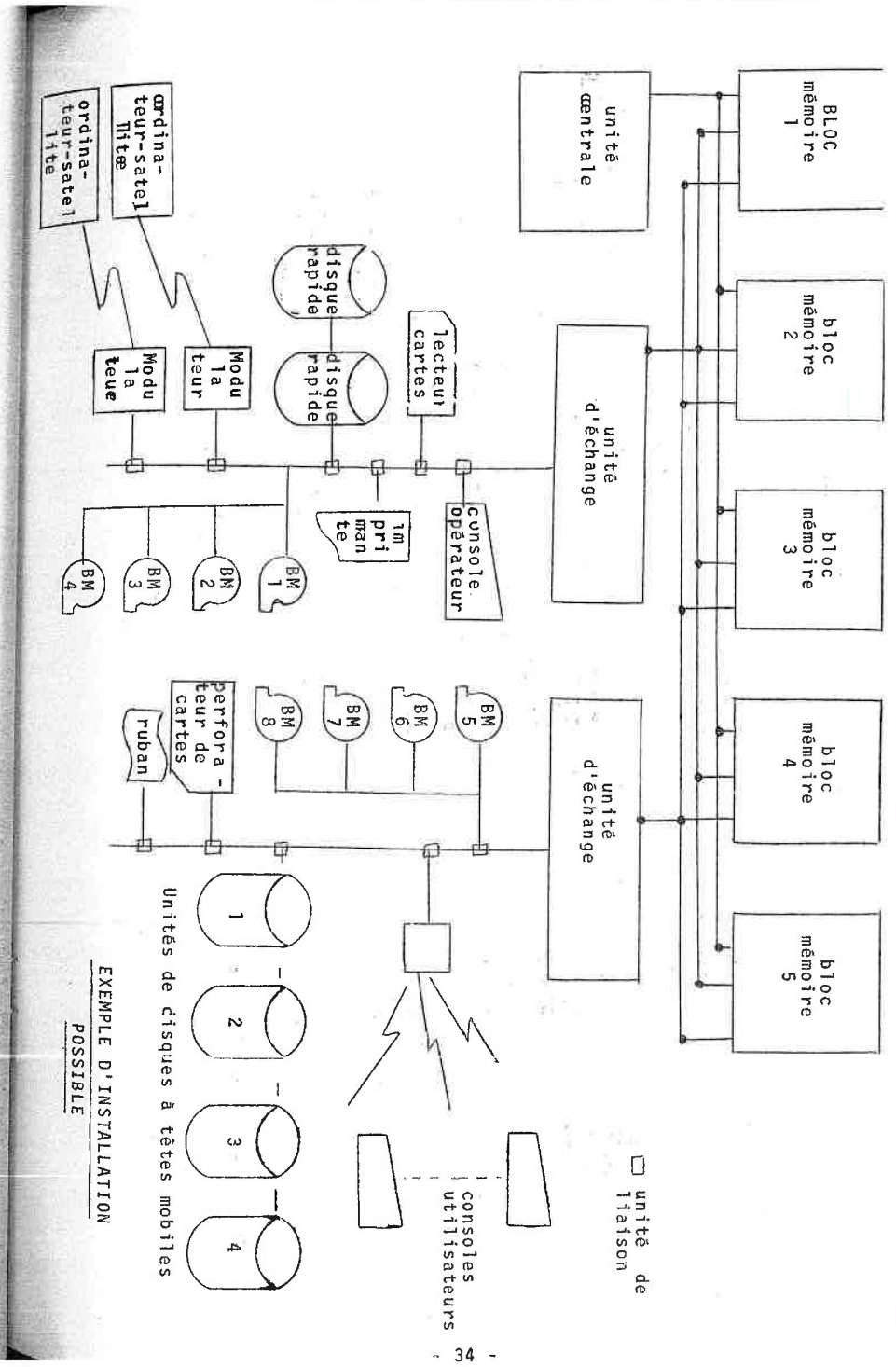
II.2.7. Les ordinateurs satellites :

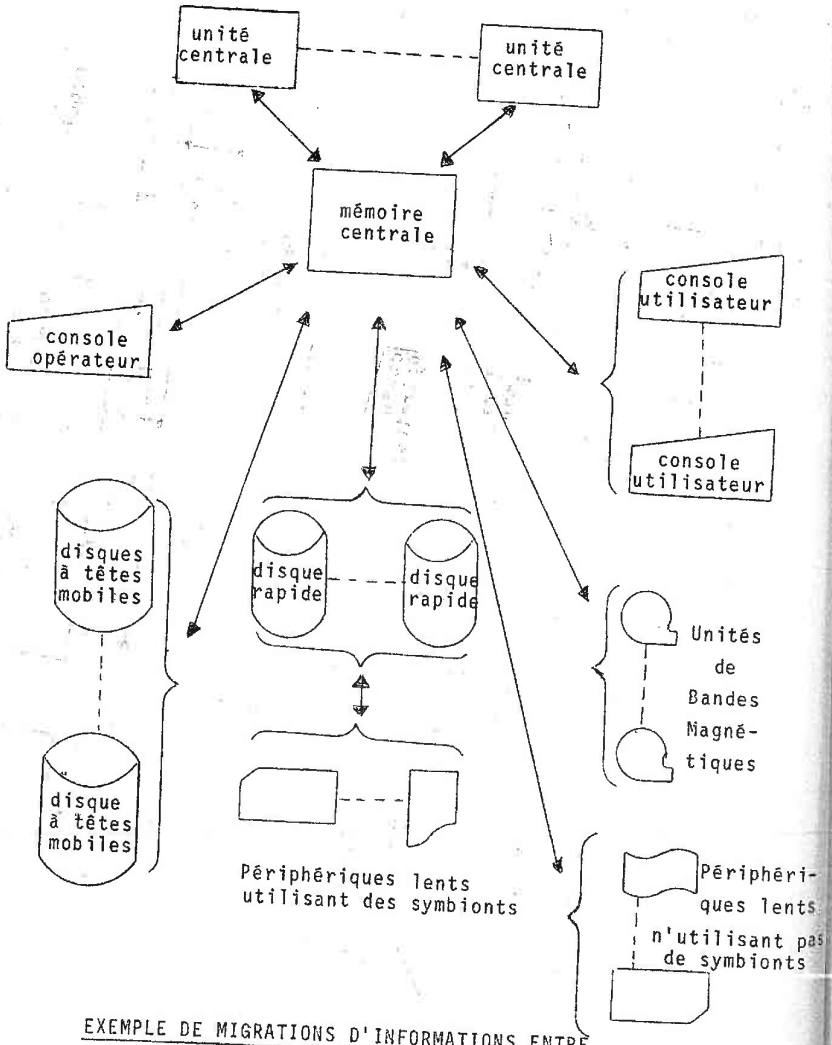
Nous nous intéresserons uniquement ici au cas d'ordinateurs satellites qui, par rapport à l'ordinateur central, ont pour fonction de lui envoyer des travaux à effectuer et

d'en recevoir des résultats.

L'ordinateur satellite est alors totalement transparent et les entrées et sorties s'effectuent de manière analogue aux entrées et sorties effectuées sur des périphériques dans le site de l'ordinateur central.

Par suite nous n'aurons pas à nous préoccuper, pour notre modèle, du fait qu'une entrée ou une sortie s'effectue sur un organe situé dans ou hors du site.





EXEMPLE DE MIGRATIONS D'INFORMATIONS ENTRE ELEMENTS DU MATERIEL

II.3. Le fonctionnement de l'installation sous moniteur de multiprogrammation :

Après avoir étudié le matériel et les paramètres probabilistes qui y sont liés, nous allons décrire son fonctionnement contrôlé par un moniteur de multiprogrammation. Nous mettrons alors en place les paramètres probabilistes qui en découlent et que nous utiliserons dans notre modèle.

II.3.1. Description générale d'un système de multiprogrammation :

L'ensemble des ressources matérielles et programmées d'une installation est géré par un moniteur qui les affecte aux travaux qui lui sont soumis.

Dans cette partie nous nous intéresserons aux systèmes de multiprogrammation. Chaque travail qu'on se propose d'effectuer est composé d'un ou plusieurs processus, ou "tâche". La caractéristique de la multiprogrammation est qu'une tâche utilisateur qui s'exécute peut être amenée à abandonner, avant sa fin, le contrôle de l'unité centrale à une autre tâche utilisateur.

Cet abandon se fera lors de l'arrivée de certains évènements, en particulier du lancement par le programme actuellement détenteur de l'unité centrale de demandes de service entraînant une attente. Il pourra se faire aussi à la suite de signaux d'horloge lorsque l'on utilise le découpage du temps, "time-slicing", ou le temps partagé, "time-sharing".

Dans ce dernier cas on a, à la fois, découpage du temps et accès direct des utilisateurs à la machine par l'intermédiaire des consoles-utilisateurs.

Par découpage du temps nous entendons le fait que l'unité centrale est attribuée aux utilisateurs par tranches de temps de durée fixée, ou quanta.

Dans un premier temps nous supposerons que nous avons une homogénéité dans le traitement des travaux, c'est-à-dire que nous pouvons avoir une exploitation de tous les travaux en temps partagé, ou en multiprogrammation avec ou sans découpage du temps, mais nous n'aurons pas de situation du type : le système est divisé en deux parties, l'une réservée à des travaux en temps partagé et l'autre à un arrière plan de travaux de type "train". Néanmoins cette hypothèse sera envisagée ultérieurement dans le paragraphe sur l'élargissement du modèle consacré aux systèmes non homogènes (cf. IV 3.4).

A partir du moment où un travail est entré dans le système on lui donne un identificateur.

Cet identificateur est d'abord placé dans la file E des travaux candidats au lancement. Puis le travail est lancé : son identificateur est alors retiré de la file E et placé dans une autre file que nous appellerons V, c'est la file des travaux qui ont été lancés, c'est-à-dire qui se sont vu attribuer le contrôle de l'unité centrale.

A partir du moment où un travail est lancé, les différentes tâches qui le composent vont pouvoir s'exécuter, lorsqu'elles auront le contrôle de l'unité centrale, les unes après les autres. Enfin le travail sera considéré comme terminé et son identificateur retiré de la file V, quand il ne demandera plus de temps de calcul ; il est à noter qu'il pourra alors être encore présent dans les fichiers-symbionts du système, jusqu'à ce que ses dernières sorties soient terminées.

Nous allons maintenant étudier successivement les files E et V, après quoi nous nous intéresserons à la gestion des entrées-sorties.

II.3.1.1. La gestion de la file E des travaux candidats au lancement :

C'est un module du moniteur, le régulateur (scheduler) qui gère la file E des travaux candidats au lancement.

Cette gestion et la structure de la file E sont définies en fonction des objectifs de base du système, par exemple : optimisation du rendement global de l'installation, ou de l'usage d'une ressource particulière, ou minimisation du temps de réponse aux utilisateurs, pour tous, ou seulement certains d'entre eux etc... De très nombreuses variantes sont donc possibles.

Nous allons étudier successivement l'entrée dans la file E, sa structure, et enfin la sortie de la file E.

II.3.1.1.1. L'entrée dans la file E :

Nous trouvons en entrée dans la file E les travaux qui viennent d'entrer dans le système.

Un travail qui entre dans le système est rangé en mémoire secondaire sous forme de fichier. On lui attribue alors un identificateur et il va devenir candidat au lancement.

L'entrée dans le système peut se faire de différentes manières, selon les installations : ainsi tous les travaux

seront entrés par un lecteur de cartes "dans le site" (accès simple), ou bien on pourra avoir de plus l'arrivée de travaux envoyés par des ordinateurs satellites, (accès multiple). Le cas d'entrées à partir de consoles utilisateurs sera étudié ultérieurement, au chapitre IV(cf.IV 3.3.).

Lorsqu'on utilise l'accès multiple, la prise en compte des travaux au moment de leur entrée dans le système se fera en fonction de priorités définies à l'avance : par exemple un travail entrant par tel organe a priorité (à l'entrée) sur celui entrant par tel autre.

Il est à noter que lorsque la file est alimentée uniquement par des travaux entrés par un lecteur de cartes, accès simple, la probabilité qu'elle soit vide pourra dépendre essentiellement des caractéristiques physiques du lecteur de cartes et de la taille, en nombre de cartes lues, des travaux entrés, par rapport à la vitesse de traitement en unité de calcul (cf III.4.).

II.3.1.1.2. La structure de la file E.

La file E pourra être divisée en plusieurs sous-files, un travail candidat au lancement étant affecté à l'une ou l'autre en fonction de l'algorithme de gestion.

Ces sous-files peuvent être définies à partir d'un grand nombre de critères ;nous en citerons quelques uns à titre d'exemple :

- demande ou non d'allocation d'une ressource rare.
- type de travail connu à partir de renseignements fournis par l'utilisateur ou établis par le système.
- plus ou moins grande urgence du travail. En liaison avec ce critère on peut envisager qu'un travail puisse changer de sous-file s'il est candidat sans succès, depuis un certain temps, au lancement.

II.3.1.1.3. La sortie de la file E.

Les sorties de la file E se font à la suite de réveils du régulateur. Le régulateur sera réveillé par la fin de l'entrée d'un travail sur mémoire secondaire, ou par l'abandon ou la demande de ressources d'un travail présent dans la file V, par l'action d'un opérateur, par un signal émis depuis une console, ou encore par un signal d'horloge etc...

Le régulateur va alors tenter le lancement d'un travail.

Pour cela chaque sous-file de E est affectée d'une priorité, la sous-file la plus prioritaire étant consultée la première.

Le régulateur va tenter d'activer un travail de la sous-file la plus prioritaire non vide. Le choix du travail dépendra de l'ordre de consultation à l'intérieur de la sous-file (premier entré-premier sorti, ou dernier entré-premier sorti, ou au hasard etc...) et aussi des ressources disponibles sur l'installation comparées à celles demandées par l'utilisateur. Ainsi si l'utilisateur demande quatre dérouleurs alors que deux seulement sont disponibles à ce moment là, les autres étant déjà affectés à d'autres travaux, le régulateur aura plusieurs solutions possibles, selon son algorithme de gestion : soit arrêter la consultation, la file E étant inchangée et le régulateur revenant à l'état dormant, soit continuer à consulter la même sous-file pour essayer de lancer un autre travail ; si aucun travail de cette sous-file ne peut être activé il pourra ou non consulter la sous-file suivante.

Certains systèmes utilisent la préemption, c'est-à-dire la possibilité de retirer temporairement à un travail déjà lancé la jouissance d'une ressource pour l'allouer à un autre travail plus prioritaire et que l'on veut lancer.

Dans ce cas, que nous étudierons dans les extensions du modèle, lors de l'arrivée d'un travail prioritaire on choisira

donc de lancer ce travail prioritaire, ce qui entraînera la suspension temporaire d'un ou plusieurs autres qui ont été lancés antérieurement.

Lorsque le régulateur a trouvé un travail pouvant être lancé, l'identificateur de ce travail quitte la file E pour être placé dans la file V des travaux lancés.

II.3.1.2. Etude de la file V.

A partir du moment où un travail entre dans la file V des travaux lancés, les différentes tâches qui le composent vont pouvoir s'exécuter en s'enchaînant les unes aux autres.

Mais à un moment plusieurs tâches issues de différents travaux présents dans la file V peuvent être candidates à l'attribution de l'unité centrale.

Cette attribution se fera elle aussi à partir des critères de gestion de l'ensemble de l'installation et est basée sur un système d'interruptions. Les différentes tâches sont placées sur des niveaux d'interruption différents,en fonction de leurs priorités relatives. Ainsi certaines tâches plus prioritaires que d'autres pourront obtenir le contrôle de l'unité centrale dès qu'elles seront aptes à lui fournir du travail et alors même qu'une autre tâche, moins prioritaire, l'utilise. Ou bien au contraire la tâche détentrice de l'unité centrale n'en perd le contrôle que lorsqu'elle ne peut plus lui fournir de travail, par exemple lorsqu'elle lance une entrée-sortie qui nécessite une attente.

La gestion des tâches nécessite l'étude des trois états que ces dernières peuvent prendre : opérationnel, prêt et en attente d'entrée-sortie, que nous allons étudier successivement .

II.3.1.2.1. Etat opérationnel d'une tâche :

Dans cet état la tâche a le contrôle de l'unité centrale : elle se déroule donc et peut lancer des entrées-

sorties en simultanéité.

Elle gardera le contrôle jusqu'au moment où :

- soit elle est interrompue et passe alors en l'état "prêt". Cette interruption est due à l'arrivée d'un événement placé sur un niveau d'interruption de priorité supérieure à celui de la tâche actuellement opérationnelle : la tâche doit donc être suspendue pour permettre le traitement de l'interruption et donc le déroulement d'une tâche plus prioritaire.

- soit elle ne peut plus fournir de travail à l'unité centrale : c'est le cas lorsqu'elle doit attendre le résultat d'une entrée-sortie pour poursuivre, elle devient alors " en attente d'entrée-sortie", ou lorsque le travail est terminé. En effet lorsqu'une tâche s'achève sans que le travail ne soit terminé l'enchaînement des tâches permet de passer à la tâche suivante du même travail :elle devient à son tour opérationnelle.

Lorsqu'un travail n'est plus candidat à l'utilisation de l'unité centrale, il envoie au moniteur un appel lui signalant qu'il arrive sur une fin normale ou anormale (erreur ou anomalie). Cet événement se produit donc lorsqu'une tâche du travail,qui sera de ce fait la dernière, est opérationnelle. L'identificateur du travail est alors retiré de la file V, mais le travail reste présent dans les fichiers de sortie du système jusqu'à ce que ses dernières impressions soient terminées.

L'intervalle de temps pendant lequel un utilisateur est opérationnel et ne cède le contrôle à aucun autre utilisateur est appelé intervalle d'exécution.

II.3.1.2.2. Etat "en attente d'entrée-sortie" d'une tâche :

La tâche reste dans cet état jusqu'à ce que l'entrée-sortie qu'elle demande soit exécutée. Plusieurs possibilités sont alors envisageables, le choix dépendant de l'optique dans laquelle est réalisé le système. Ainsi la tâche pourra être mise dans l'état "prêt" et attendre que le système d'interruption lui permette de reprendre le contrôle de

l'unité centrale et donc de devenir opérationnelle. Le problème se pose en particulier lorsqu'on utilise le découpage du temps : après une entrée-sortie, récupère-t'on ou non la fin du quantum de temps ?

Il est à noter que les possibilités de choix que nous laissons ouvertes influent non sur la nature des paramètres probabilistes que nous utiliserons dans notre modèle, mais sur leur valeur dans une application particulière, comme nous le verrons par la suite.

Il faut noter qu'une entrée-sortie telle que nous l'envisageons peut tout aussi bien être l'entrée en mémoire centrale d'un processeur nécessaire au travail (comme un compilateur ou un éditeur de liens) que l'écriture d'un enregistrement sur un fichier de l'utilisateur. Il s'agit d'échanges entre la mémoire centrale et les périphériques.

II.3.1.2.3. Etat "prêt" d'une tâche :

Une tâche prête est une tâche qui est candidate à l'attribution du contrôle de l'unité centrale à laquelle elle peut fournir du travail : elle n'attend donc que le contrôle de l'unité centrale pour devenir opérationnelle et détient les autres ressources qui lui sont nécessaires.

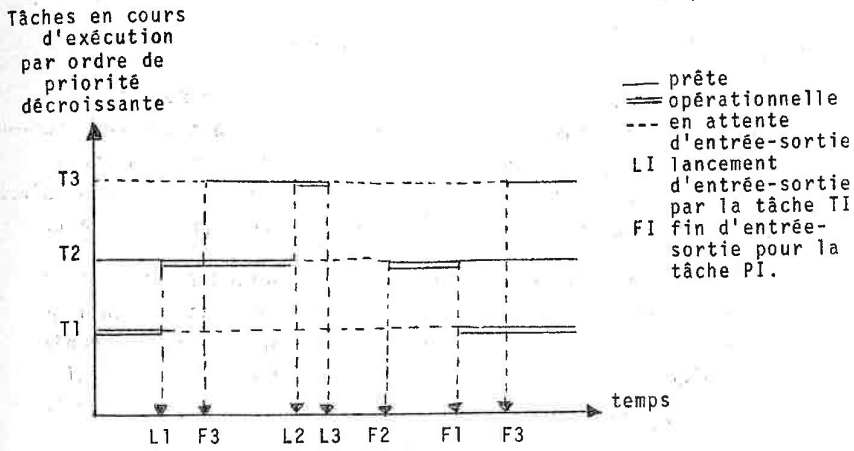
Il est à noter que, selon l'algorithme de gestion de la mémoire centrale, une tâche prête peut soit être présente totalement en mémoire, soit partiellement, ou bien ne pas y être présente.

Lorsque nous parlerons de tâches opérationnelles ou prêtes, par opposition à "en attente d'entrée-sortie" nous emploierons souvent le terme "actives".

Dans une première étape nous considérerons que, dans notre système, un travail est une suite de tâches qui doivent s'exécuter les unes à la suite des autres, un travail ne pouvant pas avoir simultanément plus d'une tâche candidate à l'activation. Par suite le nombre de travaux de la file V

est exactement le nombre de tâches-utilisateurs en cours d'exécution.

Nous verrons dans les extensions du modèle que l'on peut supprimer cette hypothèse et introduire pour les travaux la possibilité de parallélisme de plusieurs tâches.



Exemple de passage du contrôle de l'unité centrale entre les tâches en mémoire selon les priorités.

Pour conclure ces paragraphes sur la gestion des travaux dans le système nous devons insister sur le fait que de très grandes variantes existent entre systèmes, visant à l'optimisation selon certains critères et pouvant entrainer la mise en oeuvre de nombreux dispositifs. Mais en général les particularités propres à chaque installation peuvent être intégrées dans un modèle du type de celui que nous proposerons au prix, le plus souvent, de modifications minimes.

II.3.1.3. La gestion des entrées-sorties :

Les entrées-sorties faites pour le compte des utilisateurs sont prises en charge et réalisées par des modules du moniteur. L'utilisateur travaille ainsi au niveau logique (appels moniteur) sans se préoccuper de la réalisation physique (migration effective des informations d'un support à l'autre).

Les entrées-sorties des périphériques lents utilisant des symbionts se font en deux phases puisqu'elles transitent, nous l'avons vu, par les fichiers-symbionts situés en mémoire secondaire. La phase périphériques lents-mémoire secondaire est prise en charge par des modules moniteur appelés "symbionts" et qui sont des tâches parallèles du moniteur.

Le système de gestion des entrées-sorties gère donc des fichiers et des zones tampons (ou buffers) de manière à avoir une optimisation du débit global de l'installation et une simultanéité réelle entre le traitement interne et les entrées-sorties.

II.3.1.4. L'imputation du travail du moniteur :

L'exploitation d'un ordinateur en multiprogrammation entraîne la création et la mise sous contrôle du moniteur de nombreuses fonctions.

Pour cela une partie du moniteur, le noyau, réside en mémoire centrale en même temps qu'un ou plusieurs travaux utilisateurs. Le reste du moniteur est non résident ; il est rangé en mémoire secondaire : le noyau en chargera des parties en mémoire centrale quand cela sera nécessaire.

La mémoire est composée, nous l'avons vu, de L "unités d'allocation". En cours d'exploitation elles seront attribuées, de manière statique ou dynamique aux utilisateurs et au moniteur.

Dans une première phase nous ferons l'hypothèse qu'au plus ℓ peuvent être attribuées aux utilisateurs, les L-ℓ autres étant réservées au moniteur qui y range son noyau et aussi certaines parties non résidentes. Cette hypothèse nous est commode pour l'établissement de notre modèle, mais nous verrons ultérieurement qu'elle n'est en rien nécessaire : une fois encore la nature des paramètres du modèle n'en sera pas changée, mais seulement leur valeur.

Nous allons distinguer deux parties dans le travail du moniteur, selon qu'il travaille pour un utilisateur précis ou non.

. Dans le premier cas un utilisateur particulier a besoin des services du moniteur pour se poursuivre.
Par exemple le travail T002 lance un appel moniteur : le moniteur va traiter cet appel mais tout le temps de travail du moniteur (temps de calcul, entrées-sorties, etc...) sera imputé au compte de l'utilisateur T002.

. D'autres fois il n'est pas possible d'imputer à un utilisateur particulier le temps de travail du moniteur : ce sera le cas par exemple lorsque le régulateur est activé ou lorsque le moniteur doit remettre à jour ses tables etc...

Dans ce cas le travail du moniteur est imputé au compte du moniteur. Une certaine proportion du temps d'activité de l'unité centrale est donc utilisée par le moniteur pour les besoins de l'ensemble du système.

En période d'exploitation normale ce travail est composé de temps de calcul et d'échanges mémoire centrale-mémoire secondaire et mémoire centrale-console opérateur.

Pour l'écriture de notre modèle, nous ferons l'hypothèse que le moniteur, pour ses fins propres, fait, en exploitation normale, des échanges entre mémoire centrale et disques rapides et console opérateur mais qu'il n'utilise pas de disque à têtes mobiles, de bande magnétique , ou de périphérique lent.. Cette hypothèse peut très facilement être modifiée et n'a, comme nous le verrons, aucune répercussion fondamentale sur le modèle.

Mais une différence notable existe entre le moniteur et les tâches utilisateurs : le moniteur, lorsqu'il lance des demandes d'entrée -sortie, ne tombe pas en attente : il est toujours prêt à répondre à des interruptions, particulièrement aux plus prioritaires. Ceci est dû au fait que le moniteur est composé de plusieurs tâches qui, comme les tâches utilisateurs, sont placées sur différents niveaux d'interruption et ont donc différentes priorités. Certaines tâches, en particulier celles de la partie non résidente, peuvent tomber en attente, alors que d'autres, résidentes, seront toujours prêtes à prendre en considération l'arrivée d'événements les concernant. Une tâche du moniteur est dite oisive ou dormante lorsqu'elle ne s'exécute pas.

II.3.2. Les paramètres probabilistes du système :

Les développements précédents nous ont permis de préciser les caractéristiques d'une certaine classe de systèmes d'exploitation pour lesquels nous pourrons utiliser le modèle exposé au chapitre III.

Il nous faut maintenant mettre en place les paramètres probabilistes correspondants et qui nous seront nécessaires.

Nous désignerons sous le nom "d'événement" :

. les interruptions lancées par le régulateur à la suite de ses réveils.

. le passage d'une tâche d'un état à un autre, et le passage pour un travail d'une phase à l'autre de son traitement comme le passage de "candidat au lancement" à "lancé".

. les lancements et les fins d'entrées-sorties physiques, c'est-à-dire de transferts de blocs d'information d'un support à l'autre.

. le passage du contrôle d'un utilisateur au moniteur pour les besoins propres au moniteur et vice versa .

Pour notre mise en équation , nous utiliserons un intervalle de temps dt que nous supposerons suffisamment petit pour que durant cet intervalle de temps il ne puisse se produire qu'au plus un événement.

Cette hypothèse n'est pas contraignante car si, dans la réalité, deux événements arrivent au même instant, les priorités entre événements entraînent qu'ils sont pris en compte successivement.

Dès maintenant nous allons définir $P_p(t)$ comme la probabilité pour avoir à l'instant t p tâches actives, $p \leq n$ sachant que la file V comporte n éléments donc que n travaux ont chacun une tâche en cours d'exécution. $P_p(t)$ étant ainsi défini, $(1-P_0(t))$ sera donc la probabilité d'avoir au moins une tâche active à l'instant t.

Nous définirons aussi σ (respectivement $\bar{\sigma}$) comme la probabilité pour que le moniteur travaille pour lui-même lorsqu'il y a au moins une (respectivement il n'y a pas de) tâche utilisateur active. Le paramètre σ correspond donc à la mise en oeuvre de tâches du moniteur dont la priorité est plus basse que celles des tâches utilisateurs.

Nous allons d'abord étudier les paramètres qui régissent le passage des travaux de la file E à la file V puis ceux qui marquent l'évolution des tâches en mémoire.

II.3.2.1. Les paramètres de passage de la file E dans la file V.

Nous avons fait l'hypothèse, lors des définitions des files E et V, qu'un travail présent dans la file E ne peut passer dans la file V que si on peut lui allouer toutes les ressources dont il a besoin.

Chaque travail de la file E demande donc l'allocation d'un certain nombre de ressources, ces ressources sont indépendantes au sens probabiliste du terme.

Nous pouvons considérer qu'une installation dispose de plusieurs types de ressources :

. Certaines sont privées : seul un utilisateur peut y avoir accès car cette ressource lui appartient en propre. Il n'y aura pas de compétition entre travaux pour l'allocation d'une telle ressource.

. D'autres ressources sont accessibles à tous pratiquement sans limitation, et dans ce cas il n'y aura pas non plus de compétition entre travaux pour en obtenir l'allocation. Dans cette catégorie on fera entrer :

1) la plupart des ressources programmées du système.

2) les disques, si l'on suppose qu'il n'y a pas de saturation, c'est-à-dire que chacun peut se voir allouer l'espace disque qu'il désire.

3) les périphériques lents utilisant les symbionts, si l'on suppose que pour un périphérique on peut créer autant de fichiers symbionts que l'on veut sans qu'il y ait saturation de la mémoire secondaire sur laquelle ils sont écrits.

. Enfin d'autres ressources sont en quantité limitée pour l'ensemble des travaux présents dans le système. Il pourra donc y avoir "compétition" pour leur attribution. Ceci ne veut pas dire que tous les utilisateurs peuvent accéder à toutes les ressources de ce type, mais simplement qu'à un instant il peut y avoir plus de candidats que de possibilités d'accès à une ressource ou à un type de ressource. Dans cette catégorie nous trouvons des périphériques comme les dérouleurs de bandes magnétiques, des périphériques lents n'utilisant pas de symbionts, l'espace mémoire et en général les ressources ayant un nombre de points d'accès limité.

Si on veut tourner l'hypothèse de non saturation des disques on peut les faire entrer dans les ressources disponibles en quantité limitée pour les utilisateurs. Un travail ne pourra alors être lancé que s'il est assuré de disposer de tout l'espace disque qui est nécessaire à son déroulement total, y compris pour les fichiers symbionts de sortie destinés à être transcrits sur périphériques lents. La seule contrainte est alors que le système dispose pour ses besoins propres d'assez de place sur disques pour le moniteur et pour les fichiers des travaux en entrée. Si ces derniers deviennent à un moment trop importants le système peut momentanément bloquer les entrées afin que les travaux déjà présents s'écoulent. Cette hypothèse est néanmoins contraignante car elle peut aboutir à réserver plus de place que nécessaire à l'écoulement normal de l'ensemble des travaux et donc dégrader les performances du système en entraînant un freinage des entrées.

Nous ne ferons pas cette hypothèse pour l'élaboration de notre modèle et supposerons donc que les disques ne sont jamais saturés.

Nous allons donc étudier l'allocation des ressources disponibles en quantité limitée qui sont attribuées lors du lancement d'un travail.

L'ensemble des ressources en quantité limitée peut se décomposer en a classes disjointes de ressources, chaque classe contenant b_i, i = 1, ... a éléments identiques (exemple : pages mémoire, si on utilise la pagination, dérouleurs de bandes magnétiques).

Supposons que nous avons actuellement n travaux présents dans la file V. Nous noterons $\pi_{n+1}(b_i)$ la probabilité pour que n+1 travaux (les n de la file V et le travail de la file E dont la candidature est examinée) demandent au plus b_i éléments de la classe i sachant que n travaux, les n de la file V, en demandent au plus b_i.

La probabilité pour que le travail de la file E candidat à l'entrée dans la file V soit admis, c'est-à-dire puisse disposer des ressources qui lui sont nécessaires, est donc :

$$\prod_{i=1}^{a} \pi_{n+1}(b_i)$$

car toutes les ressources sont indépendantes.
Cette probabilité dépend bien sûr des caractéristiques des travaux, mais aussi de la discipline choisie pour les files E et V et des algorithmes de gestion des ressources, en particulier de l'espace mémoire. En effet la valeur des $\pi_{n+1}(b_i)$ correspondant à l'espace mémoire sera totalement différente si un travail se voit allouer au moment de son lancement et jusqu'à ce qu'il soit terminé tout l'espace mémoire nécessaire à son exécution, ou au contraire si on utilise des mécanismes d'allocation dynamique d'espace mémoire comme par exemple le chargement de page à la demande. On se reportera pour plus de détails sur cette formule au paragraphe V.B.2.3.6.

Le lancement d'un travail se fait lors d'un réveil du régulateur (cf II.3.1.1.). Nous avons vu à quelles occasions cela se produisait : nous supposerons qu'entre les instants t et t+dt il y a une probabilité λ dt d'avoir un réveil du régulateur. On peut supposer que λ au lieu d'être constant est fonction de n le nombre de travaux présents dans la file V.

II.3.2.2. Paramètres d'évolution de la file V :

Le travail qui entre dans la file V devient opérationnel, il contrôle donc l'unité centrale. Nous allons étudier les probabilités de passer d'un état à l'autre pour les différentes tâches qui vont se dérouler.

II.3.2.2.1. A partir de l'état opérationnel :

a) Passage à l'état "en attente d'entrée-sortie" :

Lorsqu'une tâche s'exécute il y a une probabilité $\omega_i$dt, i = 1 ... a, pour qu'elle lance entre t et t+dt une demande d'entrée-sortie physique sur un organe de type i.

Tout lancement d'entrée-sortie n'entraîne pas le passage à l'état "en attente d'entrée-sortie" :

. en effet si une tâche a besoin d'entrer ou de sortir de la mémoire, à un certain moment, plusieurs blocs physiques d'information (par exemple plusieurs pages du disque rapide), les demandes correspondantes ne seront pas lancées toutes "à la fois", mais successivement, de sorte que la tâche restera opérationnelle au moins jusqu'au lancement de la dernière demande.

  En fait, selon l'installation, l'enchaînement de ces différentes demandes pourra être fait par une tâche du moniteur, ou par un enchaînement des commandes sur l'unité d'échange. Mais ce qui nous importe c'est que ce travail sera imputé à l'utilisateur demandant les entrées-sorties et c'est pour cela que nous considérons qu'il reste opérationnel au moins jusqu'au lancement de la dernière demande.

. d'autre part, lorsqu'une tâche traite des fichiers séquentiels, le système de gestion des fichiers peut

anticiper et lancer la lecture du bloc physique suivant avant même que la demande ne lui ait été faite. Ceci est possible par une bonne gestion des mémoires-tampons du système.

Une demande d'entrée-sortie va entraîner avec une probabilité α_i, i = 1 ... s, le passage de la tâche à l'état "en attente d'entrée-sortie", et avec une probabilité 1 - α_i la tâche restera dans l'état opérationnel. Nous utiliserons l'indice 1 pour les disques rapides, 2 pour les disques lents et les différents types de périphériques n'utilisant pas de symbiont se verront attribuer un indice s >2 par type de périphérique.

Mais la tâche pourra aussi quitter l'état opérationnel par passage à l'état "en attente d'entrée-sortie" sans lancer simultanément une entrée-sortie physique avec une probabilité $\mu_i dt$ entre t et t+dt. Une telle situation se produit par exemple lorsqu'une tâche lance une demande d'entrée-sortie logique alors que le moniteur a déjà lancé, par anticipation, la demande d'entrée-sortie physique correspondante sur l'organe de type i, mais que cette entrée-sortie physique n'est pas encore terminée (exemple : le tampon du système n'est pas encore rempli).

b) Passage à l'état "prêt" :

L'intervalle d'exécution peut se terminer par passage à l'état prêt. C'est le cas lorsqu'on utilise la préemption de l'unité centrale par un utilisateur prioritaire ou à la fin de quanta de temps.

Cet événement a la probabilité $\mu' dt$ d'arriver entre t et t+dt. Nous ne la ferons pas intervenir pour l'établissement de notre modèle. Ce genre d'événement influe en fait sur l'avancement individuel d'un travail et non sur l'ensemble des travaux, sauf dans le cas où l'on a un chargement de la mémoire centrale selon une stratégie dynamique.

Ainsi le chargement des "pages à la demande" peut impliquer la suppression de la présence de certaines pages en mémoire de sorte que la nouvelle tâche opérationnelle peut empiéter sur la place de la précédente et donc pénaliser son déroulement ultérieur en l'obligeant à entrer à nouveau en mémoire les pages qui lui sont nécessaires (cf. II.3.2.3.2.2. l'influence de ce fait dans le modèle).

c) Passage à la fin du travail :

Enfin un utilisateur peut quitter l'état opérationnel parce qu'il a terminé son travail en mémoire centrale avec une probabilité ϕdt.

Il quitte alors la file V.

Afin de ne pas compliquer notre modèle nous envisagerons ultérieurement le cas où l'on accepte la préemption de certains organes alors qu'ils sont attribués à un travail de la file V qui en a encore besoin : le travail privé ainsi de sa ressource va devoir attendre pour se poursuivre qu'on la lui attribue à nouveau (cf. IV 2.3.).

II.3.2.2.2. A partir de l'état "en attente d'entrée-sortie" :

Passage à l'état actif :

Lorsqu'une tâche est en l'état "en attente d'entrée-sortie" elle doit attendre que l'entrée-sortie pour laquelle elle est tombée en attente soit satisfaite avant de passer à l'état prêt.

Cet évènement a des probabilités d'arriver entre t et t + dt différentes selon le périphérique utilisé pour l'entrée-sortie.

a) Disques à accès rapide :

Nous avons vu que les disques à accès rapide ont à fournir des pages :

1) aux périphériques lents utilisant les symbionts et dont les demandes arrivent, entre t et t + dt, avec une probabilité γdt.

2) aux travaux de la file V, avec une probabilité $(1-\sigma)(1-P_0(t))\,\omega_1\,dt$ entre t et t + dt : en effet c'est la probabilité d'avoir au moins une tâche active $(1-P_0(t))$, que l'unité centrale travaille pour un utilisateur et non pour les besoins du moniteur $(1-\sigma)$ et que cet utilisateur lance une demande d'entrée-sortie sur disque : $\omega_1\,dt$.

3) au moniteur pour ses besoins propres ce qui arrive entre t et t + dt avec une probabilité $\sigma(1-P_0(t))\,\omega_0\,dt + \sigma' P_0(t)\,\omega_0\,dt$.
En effet, au niveau de l'unité centrale, le temps peut être réparti en quatre catégories selon que l'unité centrale travaille ou non et selon le bénéficiaire de ce temps de travail :

. il existe en mémoire des tâches-utilisateurs actives et l'unité centrale travaille pour l'une d'elles :
probabilité au temps t : $(1-P_0(t))(1-\sigma)$

. il existe en mémoire des tâches-utilisateurs actives, mais le contrôle de l'unité centrale a été pris par le moniteur pour ses besoins propres :
probabilité au temps t : $(1-P_0(t))\,\sigma$

. il n'existe pas de tâche-utilisateur active en mémoire mais l'unité centrale travaille pour le moniteur : $\sigma' P_0(t)$

. pendant le reste du temps, soit en probabilité $(1-\sigma')P_0(t)$, l'unité centrale est oisive.
Lorsque le moniteur travaille pour ses besoins propres, il lance des entrées-sorties disque rapide avec une probabilité $\omega_0 dt$ entre t et t+dt : nous obtenons bien le résultat annoncé.

Nous supposons qu'aucune demande de transfert de page disque n'est abandonnée sans avoir été satisfaite : les transferts de pages_disques pour les différents demandeurs se font donc suivant les rapports des demandes. Ainsi la probabilité qu'un transfert soit destiné à un utilisateur sera donc le rapport des demandes des utilisateurs aux demandes totales de pages_disques, soit :

$$K = \frac{(1-\sigma)(1-P_0(t))\,\omega_1\,dt}{\gamma\,dt+(1-P_0(t))(1-\sigma)\,\omega_1\,dt+(1-P_0(t))\sigma'\omega_0\,dt+P_0(t)\,\sigma'\,\omega_0}$$

soit
$$K = \frac{(1-\sigma)(1-P_0(t))\,\omega_1}{\gamma+(1-P_0(t))(\omega_1-\sigma\omega_1+\sigma\omega_0)+P_0(t)\sigma'\omega_0}$$

Nous savons qu'en moyenne $1/\alpha_1$ demandes de page sont faites par un utilisateur avant d'avoir la fin de l'intervalle d'exécution. Une page qui transite entre le disque et la mémoire centrale pour le compte d'un utilisateur a donc une probabilité α_1 de permettre à une tâche de passer en l'état "prêt".

Pour qu'un transfert de page de disque se termine il faut que la tête de lecture passe sur la fin d'un secteur, par ex n°i, ce qui a la probabilité $\frac{dt}{\theta_1}$ d'arriver entre t et t+dt puisque le disque fait un tour en θ_1 unités de temps, il faut de plus que la file d'attente de service disque correspondant à ce secteur i ne soit pas vide : en probabilité $(1-Q^i_0(t))$. En effet nous notons $Q^i_q(t)$ la probabilité d'avoir à l'instant t q éléments dans la file correspondant au secteur n° i (cf. justification en III.2.3.).

Si nous supposons que nous avons b_1 unités de disques identiques, et pouvant travailler simultanément (cf II.2.3), chaque unité ayant la même probabilité $1/b_1$ d'être concernée lors d'une demande, nous aurons alors pour la pro-

babilité de passage, entre t et t+dt, d'une tâche de l'état "en attente d'entrée-sortie" à "active":

$$K\alpha_1 \frac{1}{\theta_1} \sum_{i=0}^{b_1 k_1} (1-Q_o^i(t))\, dt$$

ou

$$\alpha_1 K\, b_1 k_1 \theta_1^{-1} (1-Q_o^i(t))\, dt$$

En effet c'est la probabilité d'avoir entre t et t+dt la fin de transfert de la dernière page demandée par une tâche. Le deuxième résultat se déduit du premier car les files $Q^i_q(t)$ sont identiques en probabilité i.e $Q^i_q(t) = Q^j_q(t)$ $j, i \in [1, k_1 b_1]$

Remarque : Nous avons supposé qu'en cas d'utilisation de b_1 unités de disques chacune a la même probabilité d'être concernée par une demande. Cette hypothèse n'est pas nécessaire, mais elle simplifie les calculs. Si on ne la fait pas il faut étudier les files de chaque unité de disque.

b) Disques à têtes mobiles :

Contrairement aux disques à accès rapide, les disques à têtes mobiles ne font des transferts d'information que pour les utilisateurs présents dans V . Ceux-ci lancent des demandes avec une probabilité $\omega_2 dt$ entre t et t+dt, lorsqu'il y a au moins un utilisateur actif et que cet utilisateur est opérationnel (c'est-à-dire que ce n'est pas le moniteur qui travaille pour ses besoins propres), en probabilité $(1-P_o(t))(1-\sigma)$.

Et en moyenne $1/\alpha_2$ demandes seront lancées avant qu'une tâche ne tombe en attente.

Nous avons vu (cf II.2.4.) que les demandes de transferts d'information pour une unité de disque à têtes mobiles sont placées dans une file d'attente D.

La probabilité d'avoir entre t et t+dt une fin de transfert est la probabilité que la file D ne soit pas vide $(1-D_o(t))$, que de plus on passe sur une fin de secteur probabilité $k_2 dt/\theta_2$, cette fin de secteur ayant la probabilité

k'_2/k_2 de correspondre à une fin de transfert d'information puisque, nous l'avons vu, tous les secteurs ne sont pas forcément servis, pour les disques à têtes mobiles. La probabilité d'une fin de transfert d'information avec un disque à têtes mobiles entre t et t+dt sera donc :

$$(1 - D_o(t))\ k'_2 dt/\theta_2$$

Un transfert satisfait a une probabilité α_2 de permettre à une tâche de passer de "en attente d'entrée-sortie" à "active".

Si nous avons b_2 disques lents nous aurons b_2 files $D_d^{j_2}(t)$, $j_2 = 1 \ldots b_2$, que nous supposerons identiques en probabilité (cf II.2.3.).

Par suite la probabilité d'avoir le passage d'une tâche de "en attente d'entrée-sortie" à "active" entre t et t+d à la suite d'échange avec les disques lents sera :

$$(1 - D_o^{j_2}(t))\, k'_2\ \alpha_2\ b_2\ dt/\theta_2$$

c) Les périphériques autres que les disques et la console opérateur n'utilisant pas de symbionts :

Les périphériques de cette catégorie ne sont utilisés que par des travaux utilisateurs puisque nous avons supposé que le moniteur ne les utilisait pas pour ses fins propres.

Plusieurs périphériques peuvent fonctionner en même temps, mais nous avons supposé que pour chacun d'eux une demande d'entrée-sortie ne peut être lancée que si la précédente a été satisfaite. Il n'y a donc jamais d'attente de la demande : elle est traitée tout de suite.

Néanmoins, plusieurs périphériques de même type, par exemple plusieurs dérouleurs de bandes magnétiques, pouvant être attribués simultanément à un même travail et le traitement sur les périphériques pouvant se

faire en parallèle avec le traitement en unité centrale, un travail pourra lancer plusieurs demandes d'entrée-sortie sur des périphériques de même type, probabilité $\omega_s\ dt$, avant de tomber en attente : probabilité $\alpha_s\ \omega_s\ dt$, entre t et t+dt. A chaque type de périphérique correspondent des paramètres α_s et ω_s, $s > 2$.

D'autre part nous avons vu qu'une demande de transfert lancée sur un périphérique, autre que les disques et la console opérateur, n'utilisant pas de symbiont, a une probabilité $\gamma_s\ dt$ de se terminer entre t et t+dt.

Pour étudier ce phénomène, nous serons donc amenés à utiliser un cas particulier de files d'attente : chaque périphérique se voit attribuer une file d'attente ayant au plus un élément.

Nous aurons b_s files ayant chacune au maximum un élément si nous avons b_s périphériques du type numéro s pouvant travailler simultanément (cf II.2.3.).

Pour le cas de transfert sur de tels périphériques, la probabilité d'avoir le passage d'une tâche de "en attente d'entrée-sortie" à "active"sera donc :

$$b_s\ \gamma_s\ \alpha_s(1-B_0(t))\ dt$$

entre t et t + dt.

d) Les périphériques lents utilisant des symbionts :

Ces périphériques lents, nous l'avons vu, n'ont pas de rapport direct avec les tâches-utilisateurs qui se déroulent.

En effet les informations qu'ils ont à traiter transitent toujours par disques rapides et sont traitées par des tâches parallèles du moniteur, les symbionts. Ils n'ont donc aucun rôle à jouer sur les changements d'états des tâches de la file V. Ceci est le cas normal.

Nous ne considérerons pas le cas, qui ne devrait pas arriver, où la lenteur de ces périphériques pénalise le traitement en unité centrale par suite de saturation du disque si l'imprimante est trop sollicitée ou au contraire parce que des informations lues sur cartes n'ont pas pu être stockées sur disque assez vite.

e) Console opérateur :

Nous ne ferons pas intervenir la console opérateur dans le modèle que nous proposons au chapitre III pour ne pas l'alourdir, mais nous verrons dans les extensions du modèle que son introduction est possible. L'intervention de la console n'est pas nécessaire, en effet nous considérerons que les messages concernant les périphériques (exemple : demande de bande magnétique, etc...) sont inclus dans le système d'entrée-sortie et les interruptions qui s'y rapportent Les messages concernant le fonctionnement du moniteur sont entrés et sortis d'une zone de la mémoire par unité d'échange : le fonctionnement se fait donc en parallèle avec le reste du travail du moniteur et il n'y a pas lieu de faire intervenir de nouveaux paramètres.

Ceci est le cas général en fonctionnement normal et nous ne voulons pas entrer dans des cas particuliers où l'intervention du pupitreur est nécessaire pour relancer le fonctionnement de l'ensemble du système tombé en l'état oisif à la suite d'une anomalie : de toute façon on peut considérer qu'à l'échelle de temps où nous nous plaçons cet évènement a une probabilité nulle d'arriver.

II.3.2.2.3. A partir de l'état "prêt" :

Une tâche qui est à l'état prêt ne peut passer qu'en l'état opérationnel. Ceci se fera, selon les règles de priorité entre les tâches, lorsque la tâche opérationnelle perdra le contrôle. Il n'y a pas lieu pour notre modèle de faire intervenir ici de nouveaux paramètres probabilistes.

II.4. Conclusion :

Nous avons ainsi mis en place l'ensemble des paramètres probabilistes du système qui vont nous servir pour écrire les équations décrivant le comportement d'une installation satisfaisant aux hypothèses que nous avons posées.

Récapitulation des paramètres employés

a	: Nombre de ressources de types différents auxquelles peuvent avoir accès les utilisateurs.
b_i $i=1 \ldots a$	: Nombre d'éléments de la ressource i.
d	: Nombre de demandes d'entrées-sorties en attente de service au disque à têtes mobiles.
e	: Nombre de travaux candidats au lancement.
k_1	: Nombre de secteurs sur un disque rapide.
k_2	: Nombre de secteurs d'un disque à têtes mobiles.
k'_2	: Nombre de secteurs d'une unité de disque à têtes mobiles servis en un tour, en moyenne, lorsque la file d'attente de service n'est pas vide.
k	: Probabilité qu'un transfert de page disque soit destiné à une tâche utilisateur. C'est une notation.
n	: Nombre de travaux qui ont été lancés et ne sont pas terminés.
p	: Nombre de tâches actives.
q	: Nombre de demandes d'entrées-sorties en attente de service à un secteur de disque rapide.
α_i	: Probabilité qu'une demande d'entrée-sortie sur un organe de type i entraîne le passage simultané à l'état "en attente d'entrée-sortie" de la tâche qui l'a lancée.
ρdt	: Probabilité qu'un travail opérationnel se termine entre t et t+dt.
γdt	: Probabilité qu'entre t et t+dt un périphérique lent demande accès à une page disque rapide.

$\gamma = \Sigma\ \gamma_i$: γ_i est le paramètre correspondant pour le périphérique lent n°i.

$\delta_s dt$: probabilité d'avoir une fin d'entrée-sortie entre t et t+dt sur un organe de type s lorsqu'il est en service.

θ_1 :Nombre d'unités de temps pour faire un tour de disque rapide.

θ_2 :Nombre d'unités de temps pour faire un tour de disque à têtes mobiles.

λdt :Probabilité d'un réveil du régulateur entre t et t+dt.

$\mu_i dt$:Probabilité d'avoir entre t et t+dt une mise en attente d'entrée-sortie d'une tâche sans lancement simultané d'une entrée-sortie. Le passage de la tâche à l'état actif se fera à la suite d'une entrée-sortie sur un organe de type i. $\mu = \sum_i \mu_i$.

ν : $1/\nu$ est le nombre aléatoire de pages disques demandées par un utilisateur en entrée depuis un lecteur de cartes.

$\chi_n dt$:Probabilité qu'une $n^{ième}$ console utilisateur soit mise en service entre t et t+dt.

$\pi_{n+1}(b_i)$:Probabilité pour que n+1 travaux (les n travaux de la file V et le travail candidat au lancement) demandent au plus b_i éléments de la classe de ressource i sachant que n travaux (les n travaux de la file V) en demandant au plus b_i.

σ :Probabilité pour que le moniteur travaille pour lui-même lorsqu'il y a au moins une tâche utilisateur active.

σ' :Probabilité pour que le moniteur travaille pour lui-même lorsqu'il n'y a pas de tâche utilisateur active.

$\rho_i = \frac{\alpha_i\ \omega_i + \mu_i}{\omega_i}$:Probabilité pour qu'une fin d'entrée-sortie permette à une tâche en attente de devenir active.

τ :Unité de temps.

$\omega_0 dt$:Probabilité de demande d'entrée-sortie disque-rapide entre t et t+dt par le moniteur lorsqu'il travaille pour lui-même.

$\omega_i dt$:Probabilité de lancement de demande d'entrée-sortie sur un organe de type i entre t et t+dt pour un travail utilisateur.

indices : 1 disques rapides
2 disques à têtes mobiles
3 consoles utilisateurs quand elles existent sur l'installation.

CHAPITRE III

LA MISE EN PLACE DU MODELE ET SA RESOLUTION

III.1. Introduction :

Dans ce chapitre nous allons mettre en place et résoudre un modèle étudiant le comportement du système de multiprogrammation décrit au chapitre II à partir des paramètres mis en évidence.

Pour cela nous nous sommes servis particulièrement de la théorie des files d'attente (cf. Appendice).

III.2. Mise en place du modèle :

Au fur et à mesure du développement du chapitre II nous avons vu se créer de nombreuses files d'attente que nous allons étudier maintenant. Nous verrons successivement :

- la file E des travaux candidats au lancement.
- la file V des travaux qui ont été lancés et ne sont pas encore terminés.
- les files d'attente Q^i, $i = 1...k\ b_1$, des demandes de transfert de page aux $k_1 b_1$ secteurs des b_1 unités de disques rapides.
- les files d'attente D^{j_2}, $j_2 = 1...b_2$, des demandes de transfert d'information aux disques à têtes mobiles.
- les files d'attente B^{j_s}, $j_s = 1...b_s$, des demandes de transfert d'information aux b_s périphériques de type s, $s > 2$. Il s'agit ici, rappelons-le, des périphériques, autres que les disques et la console-opérateur, n'utilisant pas de symbiont.
- la file P des tâches actives.

Nous n'avons pas fait apparaître ici les consoles utilisateurs car le modèle que nous nous proposons de construire dans ce chapitre est un modèle de multiprogrammation n'utilisant pas le temps partagé.

Les différentes files sont liées les unes aux autres, et cela en particulier par le biais du paramètre n qui est le nombre de travaux présents dans la file V.

Nous étudierons $V_n(t)$, probabilité d'avoir n travaux dans la file V à l'instant t. Toutes les autres files, E, Q^i, D^{j_2}, B^{j_s} et P seront conditionnées par "il y a n travaux dans la file V".

Nous ne ferons pas apparaître ce n dans les notations pour ne pas les alourdir. Notons que dans la littérature n est souvent appelé "taux de multiprogrammation".

Nous utiliserons souvent la notation Pr pour "probabilité".

III.2.1. La file E :

Nous allons chercher $E_e(t)$, la probabilité pour qu'à l'instant t il y ait e éléments dans la file E, c'est-à-dire e travaux candidats au lancement, sachant qu'il y a n travaux dans la file V.

A cause des hypothèses posées précédemment sur dt, il peut y avoir, entre t et t+dt, au plus l'arrivée d'un nouveau travail dans E, ou le départ d'un travail de la file E, c'est-à-dire un lancement.

Les travaux entrent par un lecteur de cartes qui demande avec une probabilité $\gamma_1 dt$ une page-disque rapide, entre t et t + dt. Au moment de la lecture chaque travail demande un nombre aléatoire $1/\nu$ de pages-disque.

Par suite une entrée dans la file E a une probabilité $\nu\gamma_1 dt$ de se produire entre t et t + dt.

Une sortie de la file E correspond à une entrée dans la file V. Elle se produira s'il y a entre t et t + dt un réveil du régulateur, probabilité λ dt, qui constate qu'il peut satisfaire les demandes du candidat, probabilité $\prod_{i=1}^{a} \pi_{n+1}(b_i)$.

Par suite :

. pour e >0

$$E_e(t+dt) = E_{e+1}(t)\,[P_r \text{ une sortie de la file E entre t et t+dt}]\,[P_r \text{ pas d'entrée dans la file E entre t et t+dt}] + E_e(t)\,[P_r \text{ pas de sortie de la file E entre t et t+dt}]\,[P_r \text{ pas d'entrée dans la file E entre t et t+dt}] + E_{e-1}(t)\,[P_r \text{ pas de sortie de la file E entre t et t+dt}]\,[P_r \text{ une entrée dans la file E entre t et t+dt}]$$

. pour e = 0

$$E_0(t+dt) = E_1(t)\,[P_r \text{ une sortie de la file E entre t et t+dt}]\,[P_r \text{ pas d'entrée dans la file E entre t et t+dt}] + E_0(t)\,[P_r \text{ pas d'entrée dans la file E entre t et t+dt}]$$

Soit :

$$\begin{cases} E_e(t+dt) = E_{e+1}(t)\left[\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)dt\right]\left[1-\nu\gamma_1 dt\right] \\ \qquad + E_e(t)\left[1-\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i)dt\right]\left[1-\nu\gamma_1 dt\right] \\ \qquad + E_{e-1}(t)\left[1-\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i)dt\right]\left[\nu\gamma_1 dt\right] \\ E_0(t+dt) = E_1(t)\left[\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i)dt\right]\left[1-\nu\gamma_1 dt\right] \\ \qquad + E_0(t)\left[1-\nu\gamma_1(t)\right] \end{cases}$$

$$\Rightarrow \begin{cases} E'_e(t) = E_{e+1}(t)\left(\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i)\right) \\ \qquad + E_e(t)\left(-\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i) - \nu\gamma_1\right) \\ \qquad + E_{e-1}(t)\left(\nu\gamma_1\right) \\ E'_0(t) = E_1(t)\,\lambda\prod_{i=1}^{a}\pi_{n+1}(b_i) + E_0(t)\left[-\nu\gamma_1\right] \end{cases}$$

Ce qui, en régime permanent, c'est-à-dire pour $E'_e(t) = 0$, $e \geq 0$, nous donne :

$$E_e = \left(\frac{\nu \gamma_1}{\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)} \right)^e \left(1 - \frac{\nu \gamma_1}{\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)} \right)$$

Et nous devons avoir :

$$\nu \gamma_1 < \lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)$$

sinon la file E deviendrait infinie, ce qui n'est pas possible car la place disponible sur l'installation est finie.

III.2.2. La file V :

Nous allons étudier $V_n(t)$, probabilité qu'à l'instant t la file V ait n éléments, c'est-à-dire qu'il y ait n travaux qui ont été lancés et ne sont pas encore terminés.

Pour qu'un nouveau travail soit lancé entre t et t+dt il faut avoir un réveil du régulateur, probabilité λdt, il faut que la file E soit non vide, probabilité $(1-E_0(t))$, et que le travail candidat puisse disposer des ressources qu'il demande, probabilité $\prod_{i=1}^{a} \pi_{n+1}(b_i)$.

La probabilité de lancement sera donc $(1-E_0(t)) \cdot \prod_{i=1}^{a} \pi_{n+1}(b_i) \cdot \lambda dt$, lorsqu'il y a n travaux dans la file V.

Pour qu'un travail sorte de la file V, il faut qu'il se termine. Par suite il faut qu'il y ait une tâche-utilisateur active au moins, probabilité $(1-P_0(t))$, que cette tâche s'exécute effectivement, probabilité $(1-\sigma)$, et qu'elle se termine entre t et t+dt, probabilité βdt.

La probabilité d'avoir une sortie de la file V sera donc $(1-P_0(t))\ (1-\sigma)\ \beta dt$.

Par suite nous aurons :

$$\begin{cases} V_n(t+dt) = V_{n+1}(t) & [P_r \text{ d'une fin de travail entre t et t+dt}] \\ & [P_r \text{ pas de lancement entre t et t+dt}] \\ \quad + V_n(t) & [P_r \text{ pas de fin de travail entre t et t+dt}] \\ & [P_r \text{ pas de lancement entre t et t+dt}] \\ \quad + V_{n-1}(t) & [P_r \text{ pas de fin de travail entre t et t+dt}] \\ & [P_r \text{ un lancement de travail entre t et t+dt}] \\ \text{pour } n > 0 & \\ V_0(t+dt) = V_1(t) & [P_r \text{ d'une fin de travail entre t et t+dt}] \\ & [P_r \text{ pas de lancement entre t et t+dt}] \\ \quad + V_0(t) & [P_r \text{ pas de lancement entre t et t+dt}] \end{cases}$$

$$\Rightarrow \begin{cases} V_n(t+dt) = V_{n+1}(t) \left[1-(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_{n+2}(b_i)dt\right]\left[(1-P_0(t))\,\beta\,(1-\sigma)dt\right] \\ \quad + V_n(t) \left[1-(1-E_0(t))\,\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)dt\right]\left[1-(1-P_0(t))\,\beta\,(1-\sigma)dt\right] \\ \quad + V_{n-1}(t) \left[(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_n(b_i)dt\right]\left[1-(1-P_0(t))\,\beta(1-\sigma)dt\right] \\ V_0(t+dt) = V_1(t) \left[1-(1-E_0(t))\,\lambda \prod_{i=1}^{a} \pi_2(b_i)dt\right]\left[(1-P_0(t))\,\beta\,(1-\sigma)dt\right] \\ \quad + V_0(t) \left[1-(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_1(b_i)dt\right] \end{cases}$$

$$\Rightarrow \begin{cases} V'_n(t) = V_{n+1}(t) \left((1-P_0(t))\,\beta(1-\sigma)\right) \\ \quad + V_n(t) \left[-(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i) - (1-P_0(t))\,\beta(1-\sigma)\right] \\ \quad + V_{n-1}(t) \left((1-E_0(t))\lambda \prod_{i=1}^{a} \pi_n(b_i)\right) \\ V'_0(t) = V_1(t) \left((1-P_0(t))\,\beta(1-\sigma)\right) \\ \quad + V_0(t) \left(-(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_1(b_i)\right) \end{cases}$$

En régime permanent , donc pour $V'_n(t) = 0$, $n \geq 0$ nous avons :

$$E_0 = 1 - \frac{\nu\gamma_1}{\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)}$$

s'il y a n éléments dans la file V.

Nous pouvons donc simplifier et nous obtenons :

$$\begin{cases} 0 = V_{n+1}\,(1-P_0)\,\beta(1-\sigma) \\ \quad + V_n \left[-\nu\gamma_1 1 - (1-P_0)\,\beta(1-\sigma)\right] \\ \quad + V_{n-1}\,(\nu\gamma_1 1) \\ 0 = V_1 \quad (1-P_0)\,\beta(1-\sigma) + V_0(-\nu\gamma_1 1) \end{cases}$$

On en déduit ;

$$\boxed{V_n = \left(\frac{\nu\gamma_1}{(1-P_0)\,\beta(1-\sigma)}\right)^n \left(1 - \frac{\nu\gamma_1}{(1-P_0)\,\beta(1-\sigma)}\right)} \quad \text{pour } n \geqslant 0$$

si la relation suivante, nécessaire à la convergence, est vérifiée :

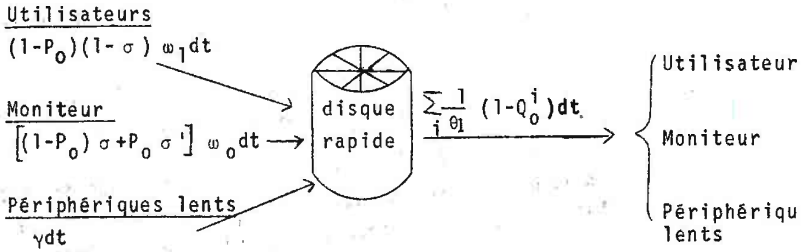
$$\boxed{\nu\gamma_1 < (1-P_0)(1-\sigma)\beta}$$

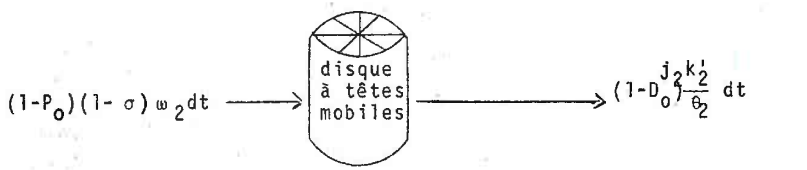
Probabilités de demande et de fin d'entrée -sortie

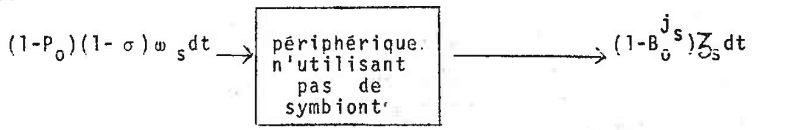
- -disque rapide
- -disque à têtes mobiles
- -autre périphérique n'utilisant pas de symbiont .

Probabilités d'arrivée de demande d'entrée -sortie

Probabilités de fin d'entrée -sortie







III.2.3. Les files Q^i :

Nous avons k files d'attente Q^i correspondant aux k secteurs du disque rapide. $Q^i_q(t)$ est la probabilité pour qu'à l'instant t la file correspondant au secteur numéro i du disque rapide ait q éléments, sachant qu'il y a n travaux dans la file V.

S'il y a b_1 unités de disques rapides de mêmes caractéristiques et travaillant en simultanéité, cela revient à considérer que l'on a $k_1 b_1$ files Q^i.

Lorsqu'une demande d'entrée-sortie est lancée à un secteur du disque, la durée du temps de service est :

. soit constante et égale à la durée d'un tour du disque ou uniforme sur $[\theta_1 - \theta_1/k_1, \theta_1]$, si la file n'était pas vide lors de l'arrivée de la demande.

. soit aléatoire si la demande arrive alors que la file est vide. Dans ce cas la durée du temps de service suit une loi uniforme sur l'intervalle $[\theta_1/k_1, \theta_1 + \theta_1/k_1]$, θ_1 étant la durée d'un tour du disque et θ_1/k_1 la durée du passage d'un secteur.

Nous avons utilisé dans notre modèle une approximation par un processus de Poisson de paramètre $1/\theta_1$, la validité de cette approximation est à étudier dans chaque cas selon la durée de rotation du disque, le nombre de secteurs et la probabilité qu'a la file d'être vide lorsqu'arrive une nouvelle demande.

Cette approximation a le très gros avantage de nous permettre d'avoir, pour l'ensemble des files d'attente de notre réseau, des arrivées poissonniennes et des temps de service distribués selon des lois exponentielles:cette propriété est très intéressante pour la résolution des équations du modèle.

La file Q^i aura un élément de plus :

. si une tâche utilisateur est opérationnelle, probabilité $(1-P_0(t))(1-\sigma)$, et lance une demande de transfert de page-disque, probabilité $\omega_1 dt$, entre t et t+dt, et que de plus cette page appartient au secteur i, probabilité $1/k_1 b_1$.

. ou si le moniteur travaille pour lui-même, probabilité $\sigma\,(1-P_0(t) + \sigma' P_0(t)$, et qu'il lance une demande de transfert pour ce secteur, probabilité $\omega_0 dt/k_1 b_1$, entre t et t+dt.

. ou si une page-disque est demandée par un module de gestion des périphériques lents utilisant des symbionts :probabilité γdt entre t et t+dt.

Nous en déduisons :

$$\begin{cases} Q^i_q(t+dt) = Q^i_{q+1}(t) & [\text{Pr d'une fin d'entrée-sortie entre t et t+dt}] \\ & [\text{Pr pas de nouvelle demande entre t et t+dt}] \\ \quad + Q^i_q(t) & [\text{Pr pas de fin d'entrée-sortie entre t et t+dt}] \\ & [\text{Pr pas de nouvelle demande entre t et t+dt}] \\ \quad + Q^i_{q-1}(t) & [\text{Pr pas de fin d'entrée-sortie entre t et t+dt}] \\ & [\text{Pr une nouvelle demande entre t et t +dt}] \\ \text{pour } q > 0 \\ Q^i_0(t+dt) = Q^i_1(t) & [\text{Pr d'une fin d'entrée-sortie entre t et t+dt}] \\ & [\text{Pr pas de nouvelle demande entre t et t+dt}] \\ \quad + Q^i_0(t) & [\text{Pr pas de nouvelle demande entre t et t+dt}] \end{cases}$$

$$
\begin{cases}
Q_q^i(t+dt) = Q_{q+1}^i(t)\left(\frac{dt}{\theta_1}\right) \\
\qquad \left(1-\left[(1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right]\frac{dt}{k_1 b_1}\right) \\
\qquad + Q_q^i(t)\left(1-\frac{dt}{\theta_1}\right) \\
\qquad \left(1-\left[(1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right]\frac{dt}{k_1 b_1}\right) \\
\qquad + Q_{q-1}^i(t)\left(1-\frac{dt}{\theta_1}\right) \\
\qquad \left((1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right)\frac{dt}{k_1 b_1} \\
Q_0^i(t+dt) = Q_1^i(t)\left(\frac{dt}{\theta_1}\right) \\
\qquad \left(1-\left[(1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right]\frac{dt}{k_1 b_1}\right) \\
\qquad + Q_0^i(t) \\
\qquad \left(1-\left[(1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right]\frac{dt}{k_1 b_1}\right)
\end{cases}
$$

$$
\begin{cases}
Q_q^{i'}(t) = Q_{q+1}^i(t)\cdot\frac{1}{\theta_1} \\
\qquad + Q_q^i(t)\left[-\frac{1}{\theta_1}-\left((1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t)\,\omega_0+\gamma\right)\cdot\frac{1}{k_1 b_1}\right] \\
\qquad + Q_{q-1}^i(t)\left[(1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\,\omega_0+\gamma\right)\cdot\frac{1}{k_1 b_1}\Big] \\
Q_0^{i'}(t) = Q_1^i(t)\cdot\frac{1}{\theta_1} \\
\qquad + Q_0^i(t)\left[-\left((1-P_0(t))(1-\sigma)\,\omega_1+(\sigma(1-P_0(t))+\sigma' P_0(t))\omega_0+\gamma\right)\cdot\frac{1}{k_1 b_1}\right]
\end{cases}
$$

En régime permanent nous obtenons donc :

$$
Q_q^i = \left[\frac{\theta_1}{k_1 b_1}\left((1-P_0)(1-\sigma)\,\omega_1+(\sigma(1-P_0)+\sigma' P_0)\,\omega_0+\gamma\right)\right]^q \left[1-\frac{\theta_1}{k_1 b_1}\left((1-P_0)(1-\sigma)\,\omega_1+(\sigma(1-P_0)+\sigma' P_0)\,\omega_0+\gamma\right)\right]
$$

$q \geq 0 \qquad i = 1 \ldots k_1 b_1$

Avec la condition nécessaire de convergence :

$$
\frac{\theta_1}{k_1 b_1}\left[(1-P_0)(1-\sigma)\omega_1+(\sigma(1-P_0)+\sigma' P_0)\,\omega_0+\gamma\right] < 1
$$

Remarque :

La formule se simplifie si l'on suppose $\omega_0 = \omega_1$, c'est-à-dire que la probabilité de demande de pages du disque est la même pour le moniteur et les utilisateurs. On obtient alors :

$$
Q_q^i = \left(\frac{\theta_1}{k_1 b_1}\left((1-P_0)\,\omega_1+\sigma' P_0\,\omega_0+\gamma\right)\right)^q \left(1-\frac{\theta_1}{k_1 b_1}\left((1-P_0)\omega_1+\sigma' P_0\omega_0+\gamma\right)\right)
$$

$q \geq 0 \qquad i = 1 \ldots k_1 b_1$

avec $\frac{\theta_1}{k_1 b_1}\left((1-P_0)\,\omega_1+\sigma' P_0\,\omega_1+\gamma\right) < 1$

III.2.4. Les files D^{j_2} :

La file $D_d^{j_2}(t)$ étudie la probabilité d'avoir, à l'instant t, d demandes de transfert d'information entre la mémoire centrale et le disque à têtes mobiles j_2 ;

$j_2 = 1 \ldots b_2$, si nous avons b_2 unités de disques à têtes mobiles identiques et travaillant simultanément.

Pour obtenir la probabilité de sortie d'un élément de la file D^{j2}, on procède de manière analogue à ce que nous avons fait pour les files Q^i. Mais, lors d'un tour de disque à têtes mobiles, en moyenne k'_2 secteurs sont servis, les disques ayant k_2 secteurs. Par suite, entre t et t+dt, la file D^{j_2} aura une probabilité $k'_2 dt / \theta_2$ de diminuer d'un élément ; comme pour les files Q^i ceci est une approximation.

Les utilisateurs peuvent seuls demander des transferts aux disques à têtes mobiles : la file s'accroîtra donc d'un élément si une tâche utilisateur est opérationnelle, probabilité $(1-P_0(t))(1-\sigma)$, et lance une demande d'entrée-sortie aux disques à têtes mobiles, probabilité $\omega_2 dt$, à destination du disque j_2, probabilité $1/b_2$. En effet on suppose que chaque unité de disque a la même probabilité que les autres d'être demandée : cette hypothèse, nous l'avons vu, n'est pas nécessaire, mais simplifie les calculs.

Ceci nous conduit aux équations:

$$
\begin{cases}
D_d^{j_2}(t+dt) = D_{d+1}^{j_2}(t) & \text{(Pr une fin de transfert entre t et t+dt)} \\
\qquad\qquad & \text{(Pr pas de nouvelle demande entre t et t+dt)} \\
\quad + D_d^{j_2}(t) & \text{(Pr pas de fin de transfert entre t et t+dt)} \\
\qquad\qquad & \text{(Pr pas de nouvelle demande entre t et t+dt)} \\
\quad + D_{d-1}^{j_2}(t) & \text{(Pr pas de fin de transfert entre t et t+dt)} \\
\qquad\qquad & \text{(Pr une nouvelle demande entre t et t+dt)}
\end{cases}
$$

pour d 0

$$
\begin{cases}
D_o^{j_2}(t+dt) = D_1^{j_2}(t) & \text{(Pr d'une fin de transfert entre t et t+dt)} \\
\qquad\qquad & \text{(Pr pas de nouvelle demande entre t et t+dt)} \\
\quad + D_o^{j_2}(t) & \text{(Pr pas de nouvelle demande entre t et t+dt)}
\end{cases}
$$

$$
\begin{cases}
D_d^{j_2}(t+dt) = D_{d+1}^{j_2}(t)\left(\frac{k'_2}{\theta_2}dt\right)\left(1-(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}dt\right) \\
\qquad + D_d^{j_2}(t)\left(1-\frac{k'_2}{\theta_2}dt\right)\left(1-(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}dt\right) \\
\qquad + D_{d-1}^{j_2}(t)\left(1-\frac{k'_2}{\theta_2}dt\right)(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}dt \\
D_o^{j_2}(t+dt) = D_1^{j_2}(t)\left(\frac{k'_2}{\theta_2}dt\right)\left(1-(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}dt\right) \\
\qquad + D_o^{j_2}(t)\left(1-(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}dt\right)
\end{cases}
$$

$$
\begin{cases}
D_d^{j'_2}(t) = D_{d+1}^{j_2}(t)\left(\frac{k'_2}{\theta_2}\right) \\
\qquad + D_d^{j_2}(t)\left(-\frac{k'_2}{\theta_2} - (1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}\right) \\
\qquad + D_{d-1}^{j_2}(t)(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2} \\
D_o^{j_2}(t) = D_1^{j_2}(t)\frac{k'_2}{\theta_2} - D_o^{j_2}(t)(1-\sigma)(1-P_0(t))\frac{\omega_2}{b_2}
\end{cases}
$$

En régime permanent nous aurons donc :

$$
D_d^{j_2} = \left(\frac{(1-\sigma)(1-P_0)\,\theta_2\,\omega_2}{k'_2 b_2}\right)^d \left(1-\frac{(1-\sigma)(1-P_0)\,\theta_2\,\omega_2}{k'_2 b_2}\right)
$$

$d \geq 0 \qquad\qquad j_2 = 1 \ldots b_2$

Avec la condition de convergence :

$$
(1-\sigma)(1-P_0)\,\theta_2\,\omega_2 < k'_2 b_2
$$

III.2.5. Les files B^{j_s} :

Les files B^{j_s}, $j_s = 1 \ldots b_s$, étudient les demandes de service aux b_s périphériques de type s pouvant travailler en simultanéité (périphériques, autres que les disques et la console-opérateur, n'utilisant pas de symbiont).

Jusqu'ici nous avons utilisé des files d'attente de longueur non bornée. Dans le cas présent nous devons limiter la taille des files B^{j_s} à 1, pour nous conformer au comportement des périphériques .

Nous n'aurons donc à étudier que $B_0^{j_s}(t)$ et $B_1^{j_s}(t)$, probabilités respectives d'avoir 0 et 1 élément dans la file B^{j_s} à l'instant t.

Lorsqu'une demande de service est faite à un périphérique, elle ne peut être faite que par le travail auquel le périphérique est alloué, et donc quand la file B^{j_s} correspondant à ce périphérique est vide.

Lorsqu'une tâche utilisateur est active, probabilité $(1-P_0(t))(1-\sigma)$, une demande d'entrée-sortie sur un périphérique de type s a une probabilité $\omega_s dt$ d'arriver entre t et t+dt, et donc $\omega_s dt/b_s$ d'être à destination du périphérique j_s, quand le périphérique est oisif.

Quand le périphérique est actif, aucune nouvelle demande ne peut lui être faite, et la demande en cours de traitement a une probabilité $\delta_s dt$ d'être satisfaite entre t et t+dt.

Par suite nous avons :

$$B_0^{j_s}(t+dt) = B_0^{j_s}(t) \quad \text{(Pr de n'avoir pas de demande entre t et t+dt)}$$
$$+ B_1^{j_s}(t) \quad \text{(Pr pas de demande entre t et t+dt)}$$
$$\text{(Pr une fin de service entre t et t+dt)}$$

$$\Rightarrow B_0^{j_s}(t+dt) = B_0^{j_s}(t)\left(1-(1-\sigma)(1-P_0(t))\,\omega_s b_s^{-1} dt\right) + B_1^{j_s}(t)\left(1-(1-\sigma)(1-P_0(t))\,\omega_s b_s^{-1} dt\right)(\delta_s dt)$$

$$\Rightarrow B_0^{j_s}{}'(t) = B_0^{j_s}(t)\left[-(1-\sigma)(1-P_0(t))\,\omega_s b_s^{-1}\right] + B_1^{j_s}\delta_s$$

Ce qui, en régime permanent, donne :

$$B_1^{j_s} = \frac{(1-\sigma)(1-P_0)\,\omega_s}{b_s \delta_s} B_0^{j_s}$$

L'équation pour $B_1^{j_s}(t+dt)$ donnerait la même chose, nous allons donc pour terminer la résolution utiliser la relation:

$$B_1^{j_s} + B_0^{j_s} = 1$$

$$B_0^{j_s} = \frac{b_s \delta_s}{b_s \delta_s + (1-\sigma)(1-P_0)\,\omega_s}$$

$$j_s = 1 \ldots b_s$$

$$B_1^{j_s} = \frac{(1-\sigma)(1-P_0)\,\omega_s}{b_s \delta_s + (1-\sigma)(1-P_0)\,\omega_s}$$

III.2.6. La file P :

La file P va nous permettre de calculer la probabilité d'avoir, à l'instant t, p tâches actives, $p \leq n$, sachant qu'il y a n travaux dans la file V. Il y a alors n-p tâches en attente d'entrée-sortie.

Le passage d'une tâche de l'état actif à l'état en attente d'entrée-sortie se fait soit sans lancement simultané d'une entrée-sortie avec une probabilité $\mu_i dt$ (le lancement ayant été fait antérieurement), soit avec lancement simultané d'une entrée-sortie. Dans ce dernier cas le lancement d'une entrée-sortie sur un organe de type i a une probabilité α_i d'entraîner le changement d'état.

Les probabilités de lancement d'entrée-sortie, entre t et t+dt, quand une tâche au moins est active et s'exécute, probabilité $(1-\sigma)$, sont :

. pour les disques à accès rapide : $\omega_1 dt$

. pour les disques à têtes mobiles : $\omega_2 dt$

. pour les autres périphériques n'utilisant pas de symbiont : $\omega_s B_0^{j_s}(t)dt$.

En effet une demande d'entrée-sortie a une probabilité $\omega_s dt$ d'être lancée sur un organe de type s entre t et t+dt. Comme b_s périphériques de type s sont en service, le périphérique j_s a une probabilité $1/b_s$ d'être concerné par cette demande. Mais il ne sera concerné que s'il est oisif, probabilité $B_0^{j_s}(t)$.

La probabilité cherchée sera donc:

$\frac{\omega_s B_0^{j_s}(t)\ dt}{b_s}$ pour le périphérique j_s et pour l'ensemble des b_s périphériques de type s on aura donc bien $\omega_s B_0^{j_s}(t)dt$.

Pour l'ensemble des périphériques, autres que les disques et la console opérateur et n'utilisant pas de symbiont, nous aurons donc :

$$\sum_{s>2} \omega_s B_0^{j_s}(t)dt$$

pour la probabilité de lancement d'entrée -sortie, entre t et t+dt.

Le passage d'une tâche de l'état "en attente d'entrée-sortie" à active se fait lors d'une fin d'entrée-sortie. Mais de même que tout lancement d'entrée-sortie n'entraînait pas le passage en attente de la tâche, de même toute fin d'entrée-sortie n'entraîne pas le passage à active. Mais comme toute tâche devenue en attente doit repasser à l'état actif

on peut dire qu'une fin d'entrée-sortie a une probabilité

$\rho_i = \frac{\mu_i + \alpha_i \omega_i}{\omega_i}$ d'entraîner le passage d'une tâche à l'état actif (cf V.3.2.3.4. pour plus de détails).

Etudions les probabilités de fin d'entrée-sortie, entre t et t+dt, pour les différents types d'organes :

. disques à accès rapide :

C'est la probabilité d'avoir une fin de service, $\frac{dt}{\theta_1}$, correspondant à une file d'attente non vide $(1-Q_0^i(t))$, $i = 1 \ldots k_1 b_1$, et que la page transférée soit destinée à une tâche utilisateur et non à un périphérique utilisant un symbiont ou au moniteur, ceci arrive avec une probabilité K en posant :

$$K = \frac{(1-\sigma)(1-P_0(t))\,\omega_1}{(1-P_0(t))(1-\sigma)\,\omega_1 + (\sigma\,(1-P_0(t)) + \sigma\,'P_0(t))\,\omega_0 + \gamma}$$

(cf. II.3.2.2.a)

Nous aurons donc finalement pour probabilité :

$$\frac{k_1 b_1 K}{\theta_1}\,(1-Q_0^i(t))dt$$

les files Q^i étant supposées toutes identiques en probabilité.

. disques à têtes mobiles :

Un transfert se termine avec une probabilité $\frac{k'_2}{\theta_2}\,dt$ si la file d'attente j_2 n'est pas vide.

Par suite la probabilité cherchée est :

$$\frac{k'_2 b_2}{\theta_2}\,(1-D_0^{j_2}(t))dt$$

les files D^{j_2} étant supposées toutes identiques en probabilité.

. Autres périphériques n'utilisant pas de symbiont :

Sur chaque unité active la probabilité d'une fin d'entrée-sortie est $\beta_s dt$.

La probabilité de fin de transfert sur l'ensemble des organes de type s est donc $\beta_s b_s B_1^{j_s}(t)dt$, puisque les b_s unités sont indépendantes et travaillent en simultanéité et que chacune a une probabilité $\beta_s B_1^{j_s}(t)dt$ de fin de transfert entre t et t+dt.

La probabilité de fin de transfert pour l'ensemble de ces organes n'utilisant de symbiont sera donc :

$$\sum_{s>2} \beta_s b_s B_1^{j_s}(t)\, dt$$

$$\begin{cases} P_p(t+dt) = P_{p+1}(t) & \text{(Pr une tâche active en moins entre t et t+dt)} \\ & \text{(Pr aucune tâche ne devient active entre t et t+dt} \\ \quad + P_p(t) & \text{(Pr aucune tâche active en moins entre t et t+dt)} \\ & \text{(Pr aucune tâche ne devient active entre t et t+dt)} \\ \quad + P_{p-1}(t) & \text{(Pr aucune tâche active en moins entre t et t+dt)} \\ & \text{(Pr une tâche devient active entre t et t+dt)} \\ \text{pour } 0 < p < n & \\ P_n(t+dt) = P_n(t) & \text{(Pr aucune tâche active en moins entre t et t+dt)} \\ \quad + P_{n-1}(t) & \text{(Pr aucune tâche mise en attente entre t et t+dt)} \\ & \text{(Pr une tâche devient active entre t et t+dt)} \\ P_0(t+dt) = P_1(t) & \text{(Pr une tâche mise en attente entre t et t+dt)} \\ & \text{(Pr aucune tâche ne devient active entre t et t+dt)} \\ \quad + P_0(t) & \text{(Pr aucune tâche ne devient active entre t et t+dt)} \end{cases}$$

Pour alléger l'écriture nous allons poser :

$$\delta(t) = (1-\sigma)\left[\alpha_1 \omega_1 + \alpha_2 \omega_2 + \sum_{s>2} \alpha_s \omega_s B_0^{j_s}(t) + \sum_{i>0} \mu_i + \beta\right]$$

$$e(t) = \nu\gamma_1 + \frac{b_1 k_1 K \varphi_1}{\theta_1}(1-Q_0^i(t)) + \frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_0^{j_2}(t)) + \sum_{s>2} \rho_s \beta_s b_s B_1^{j_s}(t$$

où $\delta(t)dt$ est donc la probabilité d'avoir entre t et t+dt le passage d'une tâche d'active à "en attente"ou à "fin de travail", probabilité $(1-\sigma)\beta$, et $\epsilon(t)$ le passage de "en attente" à active ou un lancement probabilité $(1-E_0(t))\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i) = \nu\gamma_1$.

$$\Rightarrow \begin{cases} P_p(t+dt) = P_{p+1}(t)(\delta(t)dt)(1-\epsilon(t)dt) \\ \qquad + P_p(t)(1-\delta(t)dt)(1-\epsilon(t)dt) \\ \qquad + P_{p-1}(t)(1-\delta(t)dt)(\epsilon(t)dt) \\ 0 < p < n \\ P_n(t+dt) = P_n(t)(1-\delta(t)dt) + P_{n-1}(t)(1-\delta(t)dt)(\epsilon(t)dt) \\ P_0(t+dt) = P_1(t)(\delta(t)dt)(1-\epsilon(t)dt) + P_0(t)(1-\epsilon(t)dt) \end{cases}$$

$$\Rightarrow \begin{cases} P'_p(t) = P_{p+1}(t)\,\delta(t) + P_p(t)(-\delta(t)-\epsilon(t)) + P_{p-1}(t)\,\epsilon(t) \\ 0 < p < n \\ P'_n(t) = P_n(t)(-\delta(t)) + P_{n-1}(t)(\epsilon(t)) \\ P'_0(t) = P_1(t)(\delta(t)) + P_0(t)(-\epsilon(t)) \end{cases}$$

Ce qui nous donne en régime permanent :

$$P_1 = P_0 \frac{\epsilon}{\delta} \qquad P_p = P_0 \frac{\epsilon^p}{\delta^p} \qquad 0 < p < n$$

et $P_n = P_0 \frac{\epsilon^n}{\delta^n}$

$$\Rightarrow P_p = P_0 \frac{\epsilon^p}{\delta^p} \qquad 0 \le p \le n$$

Comme $\sum_p P_p = 1$

on a $P_0 = \dfrac{1}{\sum_{p=0}^{n} \frac{\epsilon^p}{\delta^p}}$

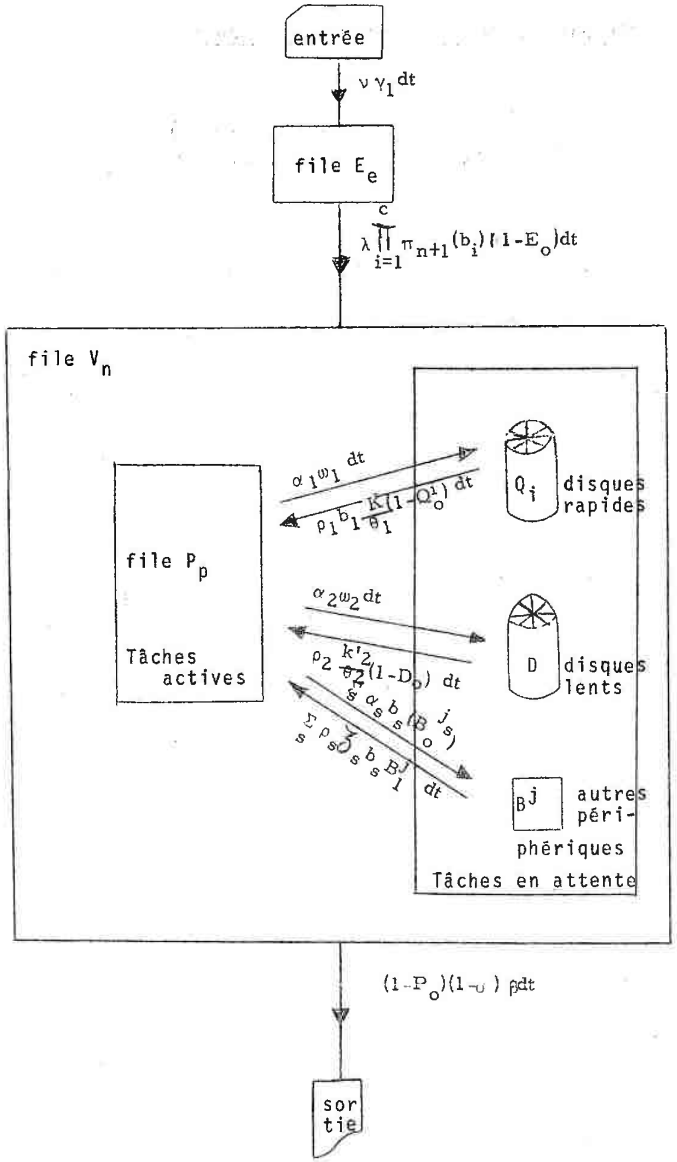
mais $\sum_{p=0}^{n} \frac{\varepsilon}{\delta}^{p} = \frac{1 - \frac{\varepsilon}{\delta}^{n+1}}{1 - \frac{\varepsilon}{\delta}}$

posons $\eta = \frac{\varepsilon}{\delta}$

c'est-à-dire :

$$\eta = \frac{\frac{f_1 b_1 k_1 K}{\theta_1}(1-Q_o^{j}) + \frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_o^{j_2}) + \sum_{s>2} \rho_s \Im_s b_s B_1^{J_s} + \nu\gamma_1}{(1-\sigma)\left[\alpha_1\omega_1 + \alpha_2\omega_2 + \sum_{s>2} \alpha_s \omega_s B_o^{J_s} + \sum_{i>0} \mu_i + \beta\right]}$$

$$P_p = \eta^p \cdot \frac{1-\eta}{1-\eta^{n+1}} \qquad 0 \le p \le n$$



Relations entre les différentes files.

II.7. Récapitulation des équations du modèle :

$$E_e = \left(\frac{\nu\gamma_1}{\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)}\right)^e \left(1 - \frac{\nu\gamma_1}{\lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)}\right) \qquad e \geq 0$$

$$V_n = \left(\frac{\nu\gamma_1}{(1-P_0)(1-\sigma)\beta}\right)^n \left(1 - \frac{\nu\gamma_1}{(1-P_0)(1-\sigma)\beta}\right) \qquad n \geq 0$$

$$Q_q^i = \left[\frac{\theta_1}{k_1 b_1}\left((1-P_0)(1-\sigma)\omega_1 + (\sigma(1-P_0) + \sigma' P_0)\omega_0 + \gamma\right)\right]^q$$

$$\left[1 - \frac{\theta_1}{k_1 b_1}\left((1-P_0)(1-\sigma)\omega_1 + (\sigma(1-P_0) + \sigma' P_0)\omega_0 + \gamma\right)\right]$$

$$i = 1 \ldots k_1 b_1 \qquad q \geq 0$$

$$D_d^{j_2} = \left(\frac{(1-\sigma)(1-P_0)\theta_2\omega_2}{k'_2 b_2}\right)^d \left(1 - \frac{(1-\sigma)(1-P_0)\theta_2\omega_2}{k'_2 b_2}\right)$$

$$j_2 = 1 \ldots b_2 \qquad d \geq 0$$

$$B_o^{j_s} = \frac{b_s \beta_s}{b_s \beta_s + (1-\sigma)(1-P_0)\omega_0} \quad ; \quad B_1^{j_s} = 1 - B_o^{j_s}$$

$$j_s = 1 \ldots b_s \qquad s > 2$$

$$P_p = \eta^p \frac{1-\eta}{1-\eta^{n+1}} \qquad 0 \leq p \leq n$$

avec

$$\eta = \frac{\frac{\rho_1 b_1 k_1 K}{\theta_1}(1 - Q_o^i) + \frac{\rho_2 k'_2 b_2}{\theta_2}(1 - D_o^{j_2}) + \sum_{s>2} \rho_s \beta_s b_s B_1^{j_s} + \nu\gamma_1}{(1-\sigma)\left(\alpha_1\omega_1 + \alpha_2\omega_2 + \sum_{s>2} \alpha_s \omega_s B_o^{j_s} + \sum_{i>0} \mu_i + \beta\right)}$$

et les conditions nécessaires :

(1) $\nu\gamma_1 < \lambda \prod_{i=1}^{a} \pi_{n+1}(b_i)$

(2) $\nu\gamma_1 < (1-P_0)(1-\sigma)\beta$

(3) $(1-P_0)(1-\sigma)\omega_1 + (\sigma(1-P_0) + \sigma' P_0)\omega_0 + \gamma < \frac{k_1 b_1}{\theta_1}$

(4) $(1-\sigma)(1-P_0)\omega_2 < \frac{k'_2 b_2}{\theta_2}$

III.3. La résolution du système.

Dans les expressions récapitulées en II.2.7., nous voyons que P_o intervient dans V_n, Q_q^i, D_d^{j2}, et B_b^{js} , b = 0,1, et que d'autre part Q_o^i, D_o^{j2} et B_o^{js} interviennent dans l'expression de P_p.

Par suite pour résoudre notre système d'équations, tous les paramètres étant supposés connus, nous devons trouver la valeur de P_o. Nous pourrons alors la reporter dans les différentes expressions et le système sera résolu.

Remplaçons donc Q_o^i, D_o^{j2} et B_1^{js} par leurs valeurs respectives dans l'expression de P_o. Nous obtenons :

$$P_o = \frac{1-\eta}{1-\eta^{n+1}} \qquad \text{avec}$$

$$\eta = \frac{\nu\gamma_1+\rho_1(1-P_o)(1-\sigma)\omega_1+\rho_2(1-P_o)(1-\sigma)\omega_2+\sum_{s>2}\rho_s\beta_s\frac{(1-\sigma)(1-P_o)\omega_s}{b_s\beta_s+(1-\sigma)(1-P_o)\omega_1}}{(1-\sigma)\left(\alpha_1\omega_1+\alpha_2\omega_2+\sum_{s>2}\alpha_s\omega_s\frac{b_s\beta_s}{b_s\beta_s+(1-\sigma)(1-P_o)\omega_s}+\sum_{i\geq 0}\mu_i+\beta\right)}$$

car $$K = \frac{(1-\sigma)(1-P_o)\omega_1}{(1-P_o)(1-\sigma)\omega_1+(\sigma(1-P_o)+\sigma' P_o)\omega_o+\gamma}$$

et $$(1-Q_o^i) = \frac{\theta_1}{k_1 b_1}\left((1-P_o)(1-\sigma)\omega_1(\sigma(1-P_o)+\sigma' P_o)\omega_o+\gamma\right)$$

$$\frac{\rho_1 b_1 k_1 K}{\theta_1}(1-Q_o^i) = \rho_1(1-\sigma)(1-P_o)\omega_1$$

et : $$1 - D_d^{j2} = \frac{(1-\sigma)(1-P_o)\theta_2\omega_2}{k'_2 b_2}$$

$$\frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_d^{j2}) = \rho_2(1-\sigma)(1-P_o)\omega_2$$

Nous avons donc à résoudre une équation du type $P_o = g(P_o)$ ou :

$$P_o = \frac{1-f(P_o)}{1-(f(P_o))^{n+1}}$$

avec

$$f(P_o) = \frac{(1-P_o)(1-\sigma)\left[\rho_1\omega_1+\rho_2\omega_2+\sum_{s>2}\rho_s\omega_s\frac{b_s\beta_s}{b_s\beta_s+(1-\sigma)(1-P_o)\omega_s}\right]+\nu}{(1-\sigma)\left(\alpha_1\omega_1+\alpha_2\omega_2+\sum_{s>2}\alpha_s\omega_s\frac{b_s\beta_s}{b_s\beta_s+(1-\sigma)(1-P_o)\omega_s}+\sum_{i\geq 0}\mu_i+\right.}$$

Cette équation se résoud par l'analyse numérique. Une fois P_o calculé on en déduit P_p, Q_q^i, D_d, B_o^j et B_1^j ainsi que V_n.

Comme E_e est déjà connu il en résulte que le système d'équations est totalement résolu en régime permanent.

III.4. Etude des conditions :

Les conditions récapitulées en III.2.8. sont les conditions nécessaires à la bonne marche du système.

Si l'une d'elles n'est pas remplie, la file d'attente correspondante deviendra très grande. Ceci entraînera un écroulement du système par saturation de l'installation, sauf si l'on a prévu des systèmes de blocage (par exemple suppression de l'entrée de nouveaux travaux) lorsque de telles situations se présentent. Dans ce dernier cas il pourra donc y avoir fin du blocage lorsque l'on estimera la situation redevenue normale. Mais si les paramètres restent inchangés, et donc si la condition nécessaire n'est toujours pas remplie on retombera dans une situation de saturation et le système ne pourra donc pas atteindre le régime permanent.

III.4.1. La condition (1) :

Dans la condition (1)

$$\nu\,\gamma_1 < \prod_{i=1}^{a} \pi_{n+1}(b_i)$$

nous voyons que pour éviter la saturation au niveau de l'entrée des travaux dans le système il faut que l'intervalle de temps moyen séparant les entrées de deux travaux dans le système $(1/\nu\gamma_1)$ soit supérieur à l'intervalle de temps moyen mis pour lancer un travail.

Une saturation à l'entrée viendra donc d'une ou de plusieurs des raisons suivantes :

.) cadence trop rapide de lecture des cartes à entrer (γ_1), surtout s'il y a plusieurs organes d'entrée travaillant simultanément (nous avons supposé que les entrées se faisaient par lecteur de cartes).

.) taille trop faible, en nombre de cartes, des travaux entrés (ν).

.) réveils trop peu fréquents du régulateur (λ).

.) manque de ressources disponibles ($\prod_{i=1}^{a} \pi_{n+1}(b_i)$).

Il est à noter, et ceci sera vrai aussi pour les autres conditions, qu'un certain nombre de paramètres varient de façon discrète, et non continue. Une modification de leur valeur pourra donc entraîner le renversement brutal du sens d'une des inégalités (1), (2), (3) ou (4).

Ainsi, si la condition (1) n'est pas remplie, une augmentation du nombre des ressources disponibles (b_i) entraînant une modification des $\pi_{n+1}(b_i)$ ne sera efficace que lorsqu'elle entraînera la réalisation de la condition (1).

On peut aussi penser qu'étant données les caractéristiques des travaux passés sur l'installation (ν), il vaut mieux avoir un lecteur de cartes moins rapide pour les entrées de travaux, (diminution de γ_1), l'économie ainsi réalisée permettant d'augmenter les ressources disponibles d'autre part (augmentation de $\pi_{n+1}(b_i)$).

III.4.2. La condition (2) :

La condition (2) :

$$\nu\gamma_1 < (1-P_0)(1-\sigma)\beta$$

met en rapport la cadence d'entrée des travaux dans le système $\nu\,\gamma_1$, avec la cadence de sortie des travaux terminés, $(1-P_0)(1-\sigma)\beta$.

S'il y a risque de saturation à ce niveau là il faut donc augmenter le temps d'activité de l'unité centrale $(1-P_0)$, et dans le temps d'activité de l'unité centrale diminuer celui consacré au moniteur (σ) ou encore il faut

diminuer le temps de calcul moyen nécessaire par travail ($1/\beta$). L'autre possibilité est de diminuer $\nu\,\gamma_1$ (cf.III.4.1.).

III.4.3. La condition (3) :

$$(1-P_0)(1-\sigma)\,\omega_1 + (\sigma(1-P_0)+\sigma' P_0)\,\omega_0 + \gamma < \frac{k_1 b_1}{\theta_1}$$

Cette condition étudie la saturation au niveau des disques rapides.

Si elle n'est pas remplie nous voyons que de nombreuses possibilités existent pour l'établir :

. Une augmentation du nombre d'unités de disque (b_1) ...

. Une augmentation du nombre de secteurs (k_1) , sans modification, a priori, de la taille des pages-disques. Ceci veut donc dire qu'on demande à accéder à une plus grande quantité d'information en un tour de disque. Une modification du nombre de secteurs ainsi que de la taille des pages aura une influence aussi sur ω_1, ω_0 et γ_1 et demande donc une étude approfondie pour déterminer dans quels cas elle est bénéfique.

. Une diminution de θ_1, le temps mis pour faire un tour de disque.

. On peut considérer a priori qu'une diminution de $(1-P_0)$, c'est-à-dire une augmentation du temps pendant lequel aucune tâche utilisateur n'est active, n'est pas bénéfique à la bonne marche de l'ensemble de l'installation.

. En revanche une diminution de l'activité du moniteur (σ et σ') pourra être envisagée, ainsi qu'une diminution de ses demandes d'entrées-sorties (ω_0).

. De même une diminution des demandes d'entrées-sorties-disques des utilisateurs devra être étudiée (ω_1).

. Ainsi qu'une diminution du trafic disque rapide-périphériques lents utilisant les symbionts (γ).

Il est à noter qu'en cas de saturation des disques rapides on peut envisager de rétablir l'inégalité (3) en supprimant totalement, ou en partie seulement, certains des termes du premier membre : ainsi on peut envisager de placer les fichiers symbionts sur bandes magnétiques et non plus sur disques rapides, ou de placer une partie des fichiers des utilisateurs sur disques à têtes mobiles, ce qui diminuera ω_1 et augmentera ω_2.

III.4.4. La condition (4) :

$$(1-\sigma)(1-P_0)\,\omega_2 < \frac{k'_2 b_2}{\theta_2}$$

La condition (4) étudie la saturation au niveau des disques à têtes mobiles. Ce qui a été dit en III.4.3. pour des disques rapides se retrouve de manière analogue pour les disques lents, il n'y a donc pas lieu de revenir sur $(1-\sigma)$, $(1-P_0)$, ω_2, b_2 et θ_2.

Pour le paramètre k'_2 nous voyons que l'augmenter signifie améliorer la gestion du disque de manière à servir plu de secteurs, en moyenne, lors d'un tour de disque.

III.4. Conclusion :

Pour conclure ce chapitre traitant de la mise en plac du modèle et de sa résolution nous pouvons dire que le mode de raisonnement basé sur l'utilisation d'un réseau de files d'attente nous a permis d'une part d'établir le comportement de l'installation en régime permanent et d'autre part de mettre en évidence des conditions entraînant l'écroulement du système par saturation, et les remèdes que l'on peut y apporter.

CHAPITRE IV

ELARGISSEMENT DU MODELE A D'AUTRES SYSTEMES

IV.I. Introduction :

Dans le chapitre II nous avons décrit les caractéristiques d'une certaine classe de systèmes de multiprogrammation et au chapitre III nous avons établi un modèle stochastique du comportement de tels systèmes.

La technique que nous avons utilisée, basée sur les files d'attente, ne s'applique pas seulement à ce cas particulier. Dans ce chapitre nous allons étudier son application d'une part à des systèmes de multiprogrammation ne satisfaisant pas à certaines hypothèses posées au chapitre II et d'autre part à d'autres types de systèmes : la monoprogrammation, le temps réel, le temps partagé, des systèmes divisés en sous-systèmes gérant de manières différentes les différents travaux qui leur sont soumis, le multitraitement.

IV.2. Extension du modèle à d'autres cas de multiprogrammation :

IV.2.1. Prise en considération de la console-opérateur :

La prise en considération de la console-opérateur ne pose pas de problème particulier, nous l'avons dit.

Cela revient à ajouter une file d'attente que nous appellerons $F_f(t)$.

Une entrée dans la file F se fera à la suite d'un lancement d'entrée-sortie sur la console soit pour le compte d'un utilisateur, probabilité $(1-P_0(t))(1-\sigma)\ \omega_1' dt$, entre t et t+dt, soit pour le compte du moniteur, probabilité $dt\ [(1-P_0(t))\,\sigma\,\omega_0' + \sigma' P_0(t)\,\omega_0'']$ entre t et t+dt. Pour le moniteur nous avons fait intervenir ω_0' et ω_0'' car nous avons considéré que la probabilité de lancement de demande d'entrée-sortie à la console pouvait changer selon que le moniteur travaille pour ses besoins propres alors qu'il y a ou non des tâches-utilisateurs actives.

Lorsqu'une demande d'entrée-sortie est en cours de traitement à la console elle a une probabilité $\beta_1' dt$ de se terminer entre t et t+dt.

Nous établirons la file F de manière analogue à ce que nous avons fait au chapitre III et nous obtiendrons en régime permanent :

$$F_f = \left(\frac{(1-P_o)(1-\sigma)\,\omega'_1+(1-P_o)\sigma\omega'_o+\sigma'P_o\omega''_o}{\zeta'_1}\right)^f \left(1-\frac{(1-P_o)(1-\sigma)\,\omega'_1+(1-P_o)\sigma\omega'_o+\sigma'P_o\omega''_o}{\zeta'_1}\right)$$

$f \geq 0$

Avec la condition nécessaire de convergence :

$$(1-P_o)(1-\sigma)\,\omega'_1 + (1-P_o)\,\sigma\omega'_o + \sigma'P_o\omega''_o < \zeta'_1$$

Nous aurons pour la console opérateur des paramètres α'_1, μ'_1 et ρ'_1 analogues aux α_i, μ_i et ρ_i établis pour les autres périphériques.

La file F interviendra donc de la même manière qu' **interviennent** les autres files des périphériques, pour l'expression de P_p.

Rappelons que :

$$P_p = \eta^p \frac{1-\eta}{1-\eta^{n+1}} \qquad 0 \leq p \leq n$$

avec $\eta = \frac{\epsilon}{\delta}$

Si l'on fait intervenir la console opérateur on ne prendra donc pas pour l'expression de P_p le paramètre η établi au chapitre III, mais :

$$\eta = \frac{\epsilon + \rho'_1\zeta'_1K'(1-F_o)}{\delta+(\alpha'\omega'_1+\mu'_1)(1-\sigma)}$$

où ϵ et δ sont les mêmes qu'au paragraphe III.2.6., c'est-à-dire

$$\delta = (1-\sigma)(\alpha_1\omega_1+\alpha_2\omega_2+\sum_{s>2}\alpha_s\omega_s B_o^{j_s}+\sum_{i>0}\mu_i+\beta)$$

$$\epsilon = \frac{\rho_1 b_1 k_1 K}{\theta_1}(1-Q_o^i)+\frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_o^{j_2})+\sum_{s>2}\rho_s\zeta_s b_s B_1^{j_s}+\nu\gamma_1$$

avec

$$K' = \frac{(1-P_o)(1-\sigma)\,\omega'_1}{(1-P_o)(1-\sigma)\omega'_1+(1-P_o)\sigma\omega'_o+\sigma'P_o\omega''_o}$$

qui représente, de manière analogue à K pour les disques rapides, la probabilité pour qu'une demande d'entrée-sortie à la console-opérateur concerne un utilisateur et non le moniteur.

La prise en considération de la console opérateur ne pose donc pas de problème particulier.

IV.2.2. La prise en compte des tâches parallèles :

Au chapitre II nous avons indiqué qu'un travail ne pouvait avoir simultanément plus d'une tâche active. Ceci nous permettait de considérer qu'il y avait autant de tâches en cours d'exécution, n, qu'il y avait de travaux dans la file V.

Dans certains systèmes il peut y avoir pour un même travail déroulement simultané de plusieurs tâches (multi-tasking).

Pour pouvoir étudier ce cas nous devons considérer que notre modèle s'intéresse non plus aux travaux mais aux tâches.

Ainsi V_n sera, dans ce cas, non plus une file de travaux lancés, mais une file de tâches qui ont été activées et ne sont pas encore terminées.

Cette file V_n aura un élément de plus, entre t et t+dt, avec une probabilité ξdt. Cet élément nouveau pourra venir soit du lancement d'un nouveau travail, soit de la mise en oeuvre pour un travail déjà lancé d'une tâche parallèle.

La file V_n aura un élément en moins lorsqu'une tâche se terminera, probabilité βdt, entre t et t+dt. Il est à noter qu'aux chapitres II et III βdt était la probabilité pour un travail de se terminer entre t et t+dt.

Nous obtiendrons donc pour la file V_n en régime permanent :

$$V_n = \left(\frac{\xi}{(1-P_o)(1-\sigma)\beta}\right)^n \left(1 - \frac{\xi}{(1-P_o)(1-\sigma)\beta}\right)$$

avec la condition de convergence :

$$\xi < (1-P_o)(1-\sigma)\beta$$

les autres files du système restant inchangées si l'on n'introduit pas d'autres modifications au comportement de l'installation (cf III.2.7.). Il y aura lieu, néanmoins, d'étudier dans ce cas l'incidence du comportement de la file E sur la file V.

IV.2.3. La préemption :

L'utilisation de la préemption dans un système revient à favoriser l'avancement de travaux particulièrement urgents, au besoin en pénalisant d'autres travaux déjà lancés

en leur retirant l'usage de ressources qui leurs sont nécessaires.

Dans notre modèle nous nous intéressons au comportement de l'installation prise dans son ensemble et non à l'avancement individuel des différents travaux.

Par suite la prise en considération de la préemption au niveau du modèle peut se borner à une modification de la valeur de certains paramètres $\pi_{n+1}(b_i)$, en particulier ceux qui concernent la mémoire secondaire ; ceci revient à considérer que les travaux suspendus par suite de la préemption neutralisent par leur présence une partie des ressources de l'installation, ce qui correspond bien à la réalité. L'utilisation de la préemption de certaines ressources peut donc entrainer un blocage par suite d'une modification de la condition (1) étudiée en III.4.2.

En fait, selon les organes sur lesquels peut porter la préemption on aura des incidences différentes sur le comportement de l'installation et il faudra établir un modèle dans chaque cas pour être fidèle à la réalité.

On pourra dans ce but établir une file d'attente S des travaux suspendus pour cause de préemption.

Mais le plus souvent le modèle établi au chapitre III suffira : il faudra simplement modifier la valeur de certains paramètres pour prendre en compte la préemption. Ceci sera vrai en particulier pour les cas de préemption de l'unité centrale ou de la mémoire centrale.

IV.2.4. Conclusion :

Nous avons ici abordé quelques modifications possibles apportées au système et pour lesquelles nous avons une modification de l'expression du modèle.

Ceci a pour but de montrer, tout comme la suite de ce chapitre, que la méthode utilisée dans l'établissement de notre

modèle stochastique permet d'étudier le comportement de systèmes très différents, le modèle devant être adapté à chaque cas particulier.

IV.3. Elargissement du modèle à d'autres systèmes :

Dans cette partie nous allons étudier comment on peut appliquer la méthode utilisée précédemment pour l'étude de systèmes très répandus : la monoprogrammation, le temps réel, le temps partagé, les systèmes d'exploitation que nous appellerons "non homogènes" et qui sont divisés en sous-systèmes gérant de manières différentes les travaux qui leur sont soumis.

IV.3.1. La monoprogrammation :

La monoprogrammation est un cas particulier du modèle que nous avons présenté au chapitre III.

Le cas de la monoprogrammation avec accès direct est sans grand intérêt pour notre modèle. Nous ne nous intéresserons donc qu'à l'accès indirect : plusieurs travaux utilisateurs peuvent être présents simultanément dans le système, mais le travail détenteur de l'unité centrale ne peut en perdre le contrôle au profit d'un autre travail utilisateur.

On appellera E la file des travaux présents dans le système, c'est-à-dire les travaux candidats au lancement et le travail lancé et on aura en régime permanent :

$$E_e = \left(\frac{\nu\gamma_1}{(1-P_o)(1-\sigma)\beta}\right)^e \left(1 - \frac{\nu\gamma_1}{(1-P_o)(1-\sigma)\beta}\right) \qquad e \geq 0$$

avec la condition : $\nu\gamma_1 < (1-P_o)(1-\sigma)\beta$

La probabilité pour qu'un travail soit en cours d'exécution est donc :

$$1-E_o = \frac{\nu\gamma_1}{\beta(1-P_o)(1-\sigma)}$$

On a donc, pour reprendre la terminologie employée dans les deux chapitres précédents :

$$\begin{cases} V_o = E_o \\ V_1 = 1 - E_o \\ V_n = 0 \qquad \forall n > 1 \end{cases}$$

La file V n'aura donc que 0 ou 1 élément, par suite $P_p = 0$ pour $p > 1$.

Les autres files sont inchangées.

IV.3.2. Le temps-réel :

En général la partie "temps réel" dans un système n'utilise pas toutes les ressources en permanence.

Aussi a-t'on coexistence dans le système de travaux temps-réel et d'un "arrière plan" d'autres travaux qui sont gérés différemment.

En pratique les travaux temps-réel sont considérés comme des modules du moniteur, souvent de très haute priorité.

Par suite pour les intégrer dans notre modèle il n'y aura donc qu'à intégrer leurs caractéristiques dans les paramètres σ, σ' et ω_o qui traduisent le fonctionnement du moniteur. Il se peut que le travail temps réel ait besoin d'autres organes : dans ce cas il n'y a aucune difficulté à faire entrer dans le modèle les autres paramètres qui le concernent, comme si c'étaient des paramètres du moniteur.

IV.3.3. Le temps partagé :

Le temps partagé suppose un accès direct des utilisateurs au système par le biais des consoles-utilisateurs.

Nous allons donc faire intervenir ici, au moyen des files C, les consoles-utilisateurs qui sont des périphé-

riques lents n'utilisant pas de symbiont (cf chapitre II).

Nous considérerons que tous les travaux traités par le système sont du même type, c'est-à-dire qu'ils sont tous lancés à partir de consoles utilisateurs. Les notions de travail et de tâche se recoupent ici et nous parlerons uniquement de tâches.

Une console ne peut avoir à un instant t plus d'une tâche en cours de traitement. Par suite le nombre de consoles en service est le nombre de tâches en cours de traitement.

Nous n'aurons donc pas de file E et nous considérerons qu'à partir du moment où une console est mise en service il se crée, de facto, une nouvelle tâche utilisateur qui est placée dans la file V.

Par suite $V_n(t)$ sera la probabilité d'avoir, au temps t, n consoles-utilisateurs en service.

Nous considérerons que les utilisateurs peuvent avoir accès, s'ils le demandent, à d'autres périphériques. Mais nous n'aurons pas, comme dans le cas exposé au chapitre III, de file d'attente des travaux candidats au lancement qui attendent pour être lancés d'avoir accès aux ressources nécessaires.

Par suite les files D^{j_2}, Q^i et B^{j_s} sont inchangées.

Nous devons donc étudier les files dont le comportement est modifié par rapport au modèle exposé au chapitre III : les files V, C et P.

Nous utiliserons l'indice 3 pour les consoles-utilisateurs.

IV.3.3.1. La file V :

La file V diminue d'un élément lorsqu'une console

est mise hors service c'est-à-dire lorsqu'un travail est terminé. Cet évènement a une probabilité $(1-P_0(t))\beta(1-\sigma)dt$ d'arriver entre t et t+dt.

La file augmentera d'un élément lorsqu'une console supplémentaire sera mise en service. Cet évènement a une probabilité $\chi_{n+1}\,dt$ d'arriver entre t et t + dt lorsqu'il y a n consoles en activité.

S'il y a b_3 consoles utilisables simultanément sur l'installation on aura donc :

$$\chi_{n+1} = 0 \qquad \text{pour} \quad n \geq b_3$$

Ceci nous permet d'écrire de manière analogue à ce que nous avons fait au chapitre III :

$$\left\{\begin{array}{l}
V_n(t+dt) = V_{n+1}(t)\left(1-\chi_{n+2}dt\right)\left(1-P_0(t)\right)(1-\sigma)\,\beta dt \\
\qquad + V_n(t)\left(1-\chi_{n+1}dt\right)\left(1-(1-P_0(t))(1-\sigma)\,\beta dt\right) \\
\qquad + V_{n-1}(t)\left(\chi_n dt\right)(1-(1-P_0(t))(1-\sigma)\,\beta dt\right) \\
\qquad \text{pour} \quad 0 < n < b_3 \\
V_0(t+dt) = V_1(t)\,(1-\chi_2 dt)(1-P_0(t))(1-\sigma)\,\beta dt \\
\qquad + V_0(t)\,(1-\chi_1 dt) \\
V_n(t+dt) = V_n(t)\,(1-(1-P_0(t))(1-\sigma)\,\beta dt) \\
\qquad + V_{n-1}(t)(\chi_n dt)(1-(1-P_0(t))(1-\sigma)\,\beta dt) \\
\qquad \text{pour} \quad n = b_3
\end{array}\right.$$

Par suite :

$$
\begin{cases}
V'_n(t) = V_{n+1}(t)\,(1-P_o(t))\,(1-\sigma)\,\beta \\
\qquad + V_n(t)\,(-\chi_{n+1} - (1-P_o(t))\,(1-\sigma)\,\beta) \\
\qquad + V_{n-1}(t)\,\chi_n \\
\qquad \text{pour } 0 < n < b_3 \\
V'_o(t) = V_1(t)\,(1-P_o(t))(1-\sigma)\,\beta \\
\qquad + V_o(t)\,(-\chi_1) \\
V'_n(t) = V_n(t)\,(-(1-P_o)(1-\sigma)\,\beta) + V_{n-1}(t)\,\chi_n \\
\qquad \text{pour } n = b_3
\end{cases}
$$

En régime permanent nous aurons donc :

$$V_n = \frac{\prod_{i=1}^{n} \chi_i}{[(1-P_o)(1-\sigma)\,\beta]^n}\, V_o$$

pour $0 < n \leq b_3$

Posons $\chi_o = 1$

$$\Rightarrow \quad V_n = \frac{\prod_{i=o}^{n} \chi_i}{(1-P_o)^n (1-\sigma)^n \beta^n} \cdot V_o \qquad 0 \leq n \leq b$$

Comme $\sum_{n=0}^{b_3} V_n = 1$

$$\Rightarrow \quad V_o \sum_{n=o}^{b_3} \left| \frac{\prod_{i=o}^{n} \chi_i}{(1-P_o)^n (1-\sigma)^n \beta^n} \right| = 1$$

$$\Rightarrow \quad V_o = \left| \sum_{n=o}^{b_3} \frac{\prod_{i=o}^{n} \chi_i}{(1-P_o)^n (1-\sigma)^n \beta^n} \right|^{-1}$$

Par suite :

$$V_n = \frac{\prod_{i=o}^{n} \chi_i}{(1-P_o)^n (1-\sigma)^n \beta^n} \left[\sum_{j=o}^{b_3} \frac{\prod_{i=o}^{j} \chi_i}{(1-P_o)^j (1-\sigma)^j \beta^j} \right]^{-1}$$

pour $0 \leq n \leq b_3$

Pour $n > b_3$: $V_n = 0$

IV.3.3.2. Les files C^{j_3} :

Le comportement des consoles sera analogue à celui des périphériques, autres que les disques et la console-opérateur, n'utilisant pas de symbiont . En effet une tâche active ne peut lancer des entrées-sorties que sur sa propre console et une nouvelle demande ne pourra être lancée que lorsque la précédente aura été satisfaite.

Nous allons donc construire des files d'attente C^{j_3} pour les consoles en service donc $0 < j_3 \leq$ de manière analogue aux files B^{j_s} : ω_3 correspondra à ω_s, ζ_3 à ζ_s et n à b_s.

Nous obtiendrons donc en régime permanent :

$$C_o^{j_3} = \frac{n\zeta_3}{n\zeta_3 + (1-\sigma)(1-P_o)\omega_3}$$

$$j_3 = 1 \ldots n \leq b_3$$

$$C_1^{j_3} = \frac{(1-\sigma)(1-P_o)\,\omega_3}{n\zeta_3 + (1-\sigma)(1-P_o)\,\omega_3}$$

IV.3.3.3. La file P :

La file P est analogue à la file P que nous avons établie au chapitre III, mais nous devons ajouter à son expression les coefficients correspondants aux entrées et sorties sur consoles-utilisateurs. Par suite nous aurons donc

$$P_p = \eta^p \quad \frac{1-\eta}{1-\eta^{n+1}} \qquad 0 \le p \le n$$

$$\eta = \frac{\frac{\rho_1 b_1 k_1 K}{\theta_1}(1-Q_o^i) + \frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_o^{j_2}) + \rho_3 \zeta_{3nC_1}^{j_3} + \sum_{s>3} \rho_s b_s \zeta_s B_1^{j_s}}{(1-\sigma)(\alpha_1\omega_1 + \alpha_2\omega_2 + \alpha_3\omega_3 C_o^{j_3} + \sum_{s>3} \alpha_s \omega_s B_o^{j_s} + \sum_{i\ge o} \mu_i + \beta)}$$

Les expressions des autres files (Q_q^i, $D_d^{j_2}$ et B^{j_s}) se trouvent en III.2.7. puisqu'elles sont inchangées.

IV.3.4. Les systèmes d'exploitation "non homogènes" :

Rappelons que nous désignons sous le terme de "systèmes d'exploitation non homogènes" les systèmes d'exploitation divisés en sous-systèmes gérant de manières différentes les travaux qui leur sont soumis.

Cette manière de procéder est souvent très intéressante pour augmenter le rendement d'une installation.

De tels systèmes se distinguent donc nettement des systèmes que nous avons étudiés jusqu'alors et qui géraient tous les travaux utilisateurs de la même manière (sauf dans le cas très particulier du temps réel étudié en IV.3.2.).

Un exemple de système non homogène est le cas où nous avons un premier plan de travaux en temps partagé et un arrière-plan en mono ou en multiprogrammation.

La répartition des organes périphériques entre les différents sous systèmes pourra être ou non fixée à l'avance.

Pour pouvoir écrire explicitement, à titre d'exemple, les équations décrivant le comportement d'un tel système nous allons poser quelques hypothèses.

Nous supposerons que nous étudions le comportement d'un système composé d'un premier plan de travaux en temps partagé issus de consoles-utilisateurs et d'un arrière-plan en multiprogrammation.

Le premier plan se voit attribuer un pourcentage $\rho^{(1)}$ du temps de calcul et l'arrière plan un pourcentage $\rho^{(2)} = (1 - \rho^{(1)})$. L'attribution se fait par tranche temps, ou quanta, de durée définie à l'avance.

Lorsqu'un sous-système ne peut pas utiliser le temps de calcul qui lui est attribué, celui-ci est utilisé par l'autre, si ce dernier peut fournir du travail à l'unité centrale.

Pendant le temps de calcul alloué à chaque sous-système on aura mis en oeuvre des travaux des utilisateurs, mais aussi des modules du moniteur.

Nous supposerons que les consoles utilisateurs sont attribuées à la partie temps partagé, ainsi qu'une partie des périphériques, et on attribuera l'indice (1) à tout ce qui concerne la partie temps partagé. L'indice (2) sera attribué à la multiprogrammation qui disposera donc d'une partie des périphériques. Nous supposerons que les disques rapides sont utilisés par les deux parties, multiprogrammation et temps partagé.

Nous allons établir les équations correspondantes en nous servant de ce que nous avons fait au chapitre III.
Nous devons changer dans les équations le $(1-P_o)$ correspondant à " il y a une tâche utilisateur active" en "il y a une tâche utilisateur de la partie (j), j = 1,2, active". Ceci correspond à la probabilité

$$(1-P_o^{(j)})\ \rho^{(j)} + \rho^{(i)}\ P_o^{(i)} \qquad i \neq j$$

Ceci est bien la probabilité qu'une tâche de la partie (j) soit active : ou bien c'est au tour de la partie (j) de se voir attribuer du temps de calcul, ou bien c'est au tour de la partie (i) mais cette dernière ne peut fournir de travail à l'unité centrale.

Nous avons supposé que le temps inutilisé par une partie pouvant être récupéré par l'autre : nous poserons donc que le moniteur travaille pour ses besoins propres une proportion $\sigma^{(j)}$ du temps attribué à la partie (j) et une proportion σ du temps lorsqu'aucune tâche utilisateur n'est active (en probabilité $(1-P_o^{(1)})\ (1-P_o^{(2)})$).

Par suite la probabilité d'avoir un lancement d'entrée-sortie au disque rapide sera :

$$\sum_{i=1}^{2} (1-P_o^{(i)})(\rho^{(i)}+\rho^{(j)}P_o^{(j)})(\omega_1^{(i)}\ (1-\sigma^{(i)})+\sigma^{(i)}\omega_o^{(i)})+(1-P_o^{(1)})(1-P_o^{(2)})\sigma\omega_o+\gamma$$

Nous noterons cette expression Ω

Lorsqu'une fin d'entrée-sortie disque rapide se produit elle a une probabilité $K^{(i)}$ d'être destinée à une tâche utilisateur de la partie (i) avec :

$$K^{(i)} = \frac{(1-P_o^{(i)})(\rho^{(i)}+\rho^{(j)}P_o^{(j)})(1-\sigma^{(i)})\omega^{(i)}}{\Omega}$$

D'autre part la probabilité d'avoir le lancement, entre t et t+dt, d'une demande d'entrée-sortie par une tâche utilisateur opérationnelle de la partie (j) sur un organe ne sera plus $\omega_s dt$ mais

$$(\rho^{(j)}+\rho^{(i)}P_o^{(i)})\,\omega_s^{(j)}\,dt$$

Les équations du modèle seront donc :

$$E_e^{(2)} = \left[\frac{\nu^{(2)}\gamma_1^{(2)}}{\lambda^{(2)}\prod_{i=1}^{a^{(2)}}\pi_{n+1}^{(2)}\ldots(b_i^{(2)})}\right]^e \left[1-\frac{\nu^{(2)}\gamma_1^{(2)}}{\lambda^{(2)}\prod_{i=1}^{a^{(2)}}\pi_{n+1}^{(2)}\ldots(b_i^{(2)})}\right]$$

$e \geq 0$

$$V_{n^{(2)}}^{(2)} = \left[\frac{\nu^{(2)}\gamma_1^{(2)}}{(1-P_o^{(2)})(\rho^{(2)}+\rho^{(1)}P_o^{(1)})(1-\sigma^{(2)})\beta^{(2)}}\right]^{n^{(2)}} \left[1-\frac{\nu^{(2)}\gamma_1^{(2)}}{(1-P_o^{(2)})(\rho^{(2)}+\rho^{(1)}P_o^{(1)})(1-\sigma^{(2)})\beta^{(2)}}\right]$$

$n \geq 0$

$$V_n^{(1)} = \frac{\prod_{i=1}^{n^{(1)}}\chi_i^{(1)}}{\left[(1-P_o^{(1)})(\rho^{(1)}+\rho^{(2)}P_o^{(2)})(1-\sigma^{(1)})\beta^{(1)}\right]^n} \left[\sum_{j=o}^{b_3^{(1)}} \frac{\prod_{i=o}^{j}\chi_i^{(1)}}{\left[(1-P_o^{(1)})(\rho^{(1)}+\rho^{(2)}P_o^{(2)})(1-\sigma^{(1)})\beta^{(1)}\right]^j}\right]^{-1}$$

pour $0 \leq n \leq b_3^{(1)}$

$$D_d^{j_2(j)} = \left[\frac{(1-\sigma^{(j)})(1-P_o^{(j)})\ (\rho^{(j)}+\rho^{(i)}P_o^{(i)})\theta_2^{(j)}\omega_2^{(j)}}{k'^{(j)}_2\ \ b_2^{(j)}}\right]^d \left[1-\frac{(1-\sigma^{(j)})(1-P_o^{(j)})(\rho^{(j)}+\rho^{(i)}P_o^{(i)})\theta_2^{(j)}\omega_2^{(j)}}{k'^{(j)}_2\ \ b_2^{(j)}}\right]$$

pour $j_2^{(j)} = 1 \ldots b_2^{(j)}$ $\qquad d \geq 0$

$i, j = 1, 2 \qquad i \neq j$

$$B_o^{j_{s(j)}} = \frac{b_s^{(j)} \zeta_s^{(j)}}{b_s^{(j)} \zeta_s^{(j)} + (1-\sigma^{(j)})(1-P_o^{(j)})(\rho^{(j)} + \rho^{(i)} P_o^{(i)})\, \omega_s^{(j)}}$$

$$B_1^{j_{s(j)}} = 1 - B_o^{j_s(j)}$$

pour $j_s = 1 \ldots b_s^{(j)}$ $\qquad i, j = 1, 2 \qquad i \neq j$

$$C_o^{j_{3(1)}} = \frac{n^{(1)} \zeta_3}{n^{(1)} \zeta_3 + (1-\sigma^{(1)})(1-P_o^{(1)})(\rho^{(1)} + \rho^{(2)} P_o^{(2)})\, \omega_3^{(1)}}$$

$j_3 = 1 \ldots n^{(1)} < b_3^{(1)}$

$$C_1^{j_{3(1)}} = 1 - C_o^{j_3(1)}$$

$$Q_q^i = \left[\frac{\theta_1}{k_1 b_1}\, \Omega\right]^q \left[1 - \frac{\theta_1}{k_1 b_1}\, \Omega\right]$$

$i = 1 \ldots k_1 b_1 \qquad q \geq 0$

$$P_{p^{(j)}}^{(j)} = (\eta^{(j)})^{p^{(j)}} \cdot \frac{1-\eta^{(j)}}{1-(\eta^{(j)})^{n^{(j)}+1}}$$

avec $0 \leq p^{(j)} \leq n^{(j)}$

Posons

$$\delta^{(j)} = (1-\sigma^{(j)})(\alpha_1^{(j)} \omega_1^{(j)} + \alpha_2^{(j)} \omega_2^{(j)} + \Sigma_{s(j)} \alpha_s^{(j)} \omega_o^{(j)} B_o^{j_s(j)} + \Sigma_i \mu_i^{(j)} + \rho^{(j)})$$

$$\epsilon^{(j)} = \frac{\rho_1^{(j)} b_1 k_1 K^{(j)}}{\theta_1} (1 - Q_o^i) + \frac{\rho_2^{(j)} k_2'^{(j)} b_2^{(j)}}{\theta_2^{(j)}} (1 - D_o^{j_2(j)}) + \Sigma_s \rho_s^{(j)} \zeta_s^{(j)} b_s^{(j)} \cdot B_1^{j_{s(j)}}$$

on aura $\eta^{(2)} = \dfrac{\epsilon^{(2)}}{\delta^{(2)}(\rho^{(2)} + \rho^{(1)} P_o^{(1)})}$

$$\eta^{(1)} = \frac{\epsilon^{(1)} + \rho_3^{(1)} \zeta_3^{(1)} n^{(1)} C_1^{j_3(1)}}{\delta^{(1)} + \alpha_3^{(1)} \omega_3^{(1)} C_o^{j_3(1)} (\rho^{(1)} + \rho^{(2)} P_o^{(2)})}$$

Avec les conditions de convergence :

$$\nu^{(2)} \gamma_1^{(2)} < \lambda^{(2)} \prod_{i=1}^{a^{(2)}} \pi_{n+1}^{(2)} \, (b_i^{(2)})$$

$$\nu^{(2)} \gamma_1^{(2)} < (1-P_o^{(2)})(\rho^{(2)} + \rho^{(1)} P_o^{(1)})(1-\sigma^{(2)}) q^{(2)}$$

$$(1-\sigma^{(j)})(1-P_o^{(j)})(\rho^{(j)} + \rho^{(i)} P_o^{(i)})\, \theta_2^{(j)} \omega_2^{(j)} < b_2^{(j)} k_2^{(j)}$$

$j = 1, 2 \qquad i \neq j$

$\theta_1 \Omega < k_1 b_1 \qquad$ (cf Ω p 107)

Les conditions de convergence s'établissent à partir des équations du modèle de manière analogue à ce que nous avons fait au chapitre III.

La résolution des équations se fera de manière analogue à ce que nous avons fait au chapitre III, mais au lieu de pouvoir tout exprimer en fonction de P_o nous exprimerons tout en fonction de $P_o^{(1)}$ et $P_o^{(2)}$. Il faudra donc résoudre

$$\begin{cases} P_o^{(1)} = f\,(P_o^{(1)}, P_o^{(2)}) \\ P_o^{(2)} = g\,(P_o^{(1)}, P_o^{(2)}) \end{cases}$$

pour obtenir la solution de l'ensemble des équations.

IV.3.5. Le multitraitement :

Nous allons étudier ici le cas où une installation dispose de plusieurs unités centrales, que nous supposerons identiques et qui travaillent en parallèle.

Nous supposerons que l'ensemble de l'installation fonctionne de manière analogue à ce que nous avons établi au chapitre II : la seule différence est qu'au lieu d'avoir une seule tâche active à la fois, il pourra y en avoir plusieurs, autant que d'unités centrales.

D'autre part comme l'utilisation du multitraitement permet d'introduire dans les travaux des tâches parallèles, nous procéderons comme en IV.2.2. : la file V sera une file de tâches et nous ne nous préoccuperons pas de la méthode employée pour introduire des tâches dans le système.

Soit n le nombre d'unités centrales pouvant travailler simultanément.

Dans le modèle à une seule unité centrale nous utilisions $(1-P_0(t) = \sum_{i=1}^{n} F_i(t)$ qui était la probabilité d'avoir une tâche-utilisateur active, cette probabilité conditionnant l'arrivée d'un certain nombre d'évènements.

Dans le cas où nous avons plusieurs unités centrales, il faut remplacer ce facteur $(1-P_0(t))$ dans les expressions où il apparait par le facteur correspondant qui est

$$\sum_{i=1}^{u} iP_i(t) \qquad \text{si } u \leq n$$

ou

$$\sum_{i=1}^{u-1} iP_i(t) + u \sum_{i=u}^{n} P_i(t)$$

De manière analogue à ce que nous avons fait au chapitre III notons εdt la probabilité d'avoir entre t et t+dt le passage en attente d'une tâche active et δdt la probabilité d'avoir entre t et t+dt le passage à l'état actif d'une tâche en attente.

Nous aurons alors pour $u < n$

$$\begin{aligned} P_0(t+dt) &= P_1(t) \quad (\delta\, dt)(1-\varepsilon dt) \\ &+ P_0(t) \quad (1-\varepsilon dt) \end{aligned}$$

$$\begin{aligned} P_p(t+dt) &= P_{p+1}(t) \ ((p+1)\,\delta\, dt)(1-\varepsilon\, dt) \\ &+ P_p(t) \quad (1-p\,\delta\, dt)(1-\varepsilon dt) \\ &+ P_{p-1}(t) \ (1-(p-1)\,\delta dt)\,\varepsilon dt \end{aligned}$$

pour $p < n$

$$\begin{aligned} P_p(t+dt) &= P_{p+1}(t) \ (u\ \delta dt)(1-\varepsilon\, dt) \\ &+ P_p(t) \quad (1-u\ \delta dt)(1-\varepsilon dt) \\ &+ P_{p-1}(t) \ (1-(u-1)\delta dt)\,\varepsilon\, dt \end{aligned}$$

pour $p = u$

$$\begin{aligned} P_p(t+dt) &= P_{p+1}(t) \ (u\ \delta dt)(1-\varepsilon\, dt) \\ &+ P_p(t) \quad (1-u\ \delta dt)(1-\varepsilon dt) \\ &+ P_{p-1}(t) \ (1-u\ \delta dt) \quad \varepsilon dt \end{aligned}$$

pour $u < p < n$

$$\begin{aligned} P_p(t+dt) &= P_n(t) \quad (1-u\ \delta dt) \\ &+ P_{n-1}(t) \ (1-u\ \delta dt)\ \varepsilon dt \end{aligned}$$

On en déduit en régime stationnaire que :

$$P_p = P_o \frac{\varepsilon^p}{\delta^p} \cdot \frac{1}{p!} \quad \text{pour } 0 \le p < u$$

$$P_p = P_o \frac{\varepsilon^p}{\delta^p} \cdot \frac{1}{u!} \cdot \frac{1}{u^{p-u}} \quad \text{pour } u \le p \le n$$

Dans ce calcul nous avons supposé ε et δ indépendants du nombre d'unités centrales actives. Si l'on veut faire intervenir le fait que le temps de calcul consacré par le moniteur à ses propres besoins peut être fonction du nombre d'unités centrales actives il faudra utiliser non pas ε mais $\varepsilon(i)$ car il n'y aura pas σ mais $\sigma(i)$ dans l'expression de ε (cf. III.2.7). Nous aurons donc :

$$P_p = P_o \frac{\prod_{i=1}^{u} \varepsilon(i)}{\delta^p} \cdot \frac{1}{p!} \quad \text{pour } 0 \le p < u$$

$$P_p = P_o \frac{\prod_{i=1}^{u} \varepsilon(i)}{\delta^p} \cdot \frac{\varepsilon^{p-u}}{u!u^{p-u}} \quad \text{pour } u \le p \le n$$

Si on a $u > n$ les résultats s'établissent de manière analogue et l'on aura :

$$P_p = P_o \frac{\varepsilon^p}{\delta^p} \cdot \frac{1}{p!} \quad \text{pour } 0 \le p \le n$$

ou

$$P_p = P_o \frac{\prod_{i=1}^{p} \varepsilon(i)}{\delta^p} \cdot \frac{1}{p!}$$

si on considère que ε dépend du nombre d'unités centrales actives.

Nous noterons

$$\rho(p_o) \begin{cases} \sum_{i=1}^{u} i P_i(t) & \text{si } u \le n \\ \sum_{i=1}^{u-1} i P_i(t) + u \sum_{i=u}^{n} P_i(t) & \text{si } u \ge n \end{cases}$$

Nous avons établi les équations régissant le comportement de l'installation dans le cas où σ est indépendant du nombre d'unités centrales actives :

$$V_n = \left[\frac{\xi}{\rho(p_o)(1-\sigma)\beta}\right]^n \left[1 - \frac{\xi}{\rho(p_o)(1-\sigma)\beta}\right] \qquad n \ge 0$$

avec $\xi < \rho(p_o)(1-\sigma)\beta$

$$Q_q^i = \left[\frac{\theta_1}{k_1 b_1}\left(\rho(p_o)(1-\sigma)\omega_1 + (\sigma\rho(p_o) + \sigma'(1-\rho(p_o))\omega_o + \gamma\right)\right]^q \left[1 - \frac{\theta_1}{k_1 b_1}\left(\rho(p_o)(1-\sigma)\omega_1 + (\sigma\rho(p_o) + \sigma'(1-\rho(p_o))\omega_o + \gamma\right)\right]$$

$$i = 1 \ldots k_1 b_1 \qquad q \ge 0$$

avec $\theta_1 \rho(p_o)(1-\sigma)\omega_1 + (\sigma\rho(p_o) + \sigma'(1-\rho(p_o))\omega_o + \gamma) < k_1 b_1$

$$D_d^{j_2} = \left[\frac{(1-\sigma)\rho(p_o)\theta_2\omega_2}{k'_2 b_2}\right]^d \left[1 - \frac{(1-\sigma)\rho(p_o)\theta_2\omega_2}{k'_2 b_2}\right]$$

$$j_2 = 1 \ldots b_2 \qquad d \ge 0$$

avec $(1-\sigma)\rho(p_o)\theta_2\omega_2 < k'_2 b_2$

$$B_o^{j_s} = \frac{b_s \zeta_s}{b_s \zeta_s + (1-\sigma)\rho(p_o)\omega_s}$$

$$b_1^{j_s} = 1 - B_o^{j_s} \qquad j_s = 1 \ldots b_s \qquad s > 2$$

$$P_p = \begin{cases} P_o \dfrac{\varepsilon^p}{\delta^p} \cdot \dfrac{1}{p!} & \text{pour } 0 \le p \le u \le n \\ & \text{et } 0 \le p \le n \le u \\ P_o \dfrac{\varepsilon^p}{\delta^p} \cdot \dfrac{1}{u!} \cdot \dfrac{1}{u^{p-u}} & \text{pour } u \le p \le n \end{cases}$$

avec :

$$\delta = (1-\sigma)(\alpha_1 \omega_1 + \alpha_2 \omega_2 + \sum_{s>2} \alpha_s \omega_s B_o^{j_s} + \sum_{i>o} \mu_i + \beta)$$

$$\varepsilon = \frac{\rho_1 b_1 k_1 K}{\theta_1}(1-Q_o^{i}) + \frac{\rho_2 k'_2 b_2}{\theta_2}(1-D_o^{j_2}) + \sum_{s>2} \rho_s \gamma_s b_s B_1^{j_s} + \xi$$

Ces équations se résolvent très facilement lorsqu'on a calculé P_o.

IV.4. Conclusion :

Les quelques cas traités dans ce chapitre nous montrent que les réseaux de files d'attente peuvent être utilisés pour l'élaboration de modèles stochastiques de systèmes d'exploitation aux caractéristiques très variées. L'intérêt de cette technique vient de sa souplesse qui lui permet de s'adapter à chaque cas particulier.

Nous n'avons pas abordé ici l'étude d'un modèle stochastique de réseau d'ordinateurs, mais nous pensons que là aussi l'utilisation d'un réseau de files d'attente pourra être très utile.

CHAPITRE V

L'EVALUATION DES PARAMETRES DU MODELE

V.1. Introduction.

L'application pratique du modèle ne peut se faire que si l'on a auparavant évalué les paramètres nécessaires.

Deux possibilités d'évaluation des paramètres existent

. L'évaluation "á postériori" sur une installation en cours de fonctionnement : grâce à des mesures "bien choisies" on met en évidence les paramètres intéressants.

. L'évaluation "à priori" : dans les cas où l'évaluation "a postériori" n'est pas possible on devra, à partir d'un certain nombre d'éléments, estimer la valeurde tous les paramètres ou simplement d'une partie d'entre eux.

Bien entendu les résultats de ces deux méthodes sont totalement différents.

En effet dans le premier cas l'application du modèle se réfère à une situation existant réellement et l'on pourra confronter les résultats réels et le modèle.

Au contraire, dans le second cas, il faudra estimer une partie au moins des paramètres utilisés, ce qui n'est pas sans poser de problèmes comme nous allons le voir.

Les différences sont liées au pourquoi de l'utilisation du modèle. Va-t-il être utilisé de manière "statique", c'est-à-dire pour étudier le comportement du système à un certain moment et éventuellement le comparer avec d'autres systèmes qui peuvent exister sur d'autres installations ?
Ou va-t-il être utilisé de manière"dynamique"? il s'agira alors, à partir d'une évaluation, de voir l'impact de certaines modifications du hardware, du software ou de la charge que l'on envisage, ou d'essayer d'optimiser le système selon certains critères.

V.2. Evaluation "a priori" des paramètres.

L'utilisation statique se réfère à l'évaluation des paramètres telle que nous la verrons ultérieurement. Au contraire, pour l'utilisation dynamique du modèle, nous allons voir que nos paramètres, loin d'être fixes et immuables, changent au fur et à mesure des modifications du système : ils ne sont pas figés, mais caractéristiques d'une installation à un certain moment. A priori, on ne connait pas les relations qui les lient, surtout dans la mesure où on fait intervenir le comportement des utilisateurs. Mais des études et modèles locaux peuvent apporter sur ce point des informations très précieuses.

Nous allons illustrer cela par quelques exemples.

Il est quelquefois possible de faire une modification sur une installation et de faire marcher le système modifié pendant "assez" longtemps pour qu'une nouvelle évaluation donne des résultats "permanents" qui supportent donc la comparaison avec les résultats antérieurs à la modification. Le "assez" dépend de ce qui est modifié : si cela joue sur le comportement des utilisateurs il leur faut le temps de s'adapter. Nous sommes, bien sûr, dans le cas d'une utilisation statique du modèle, les paramètres nécessaires étant évalués deux fois successivement.

Mais le plus souvent lorsqu'on envisage une modification, il faut a priori estimer son impact, c'est-à-dire voir ses conséquences sur les différents paramètres.

Ainsi une modification de la politique d'entrée des travaux en mémoire défavorable aux travaux gros demandeurs d'espace-disque entraînera sans doute un usage plus important des bandes magnétiques.

Ou encore une diminution de l'espace mémoire accessible aux utilisateurs entraînera une modification de la taille et de la structure des programmes utilisateurs, et ceci, comme l'a prouvé l'expérience, même si cette diminution est ignorée des utilisateurs.

Ces changements de comportement ne sont pas immédiats et leur temps de réponse est plus ou moins long selon que la modification gêne plus ou moins les utilisateurs, mais ils sont obligatoires. En effet supposons que les utilisateurs gros demandeurs d'une certaine ressource soient, à la suite d'une modification du système, gravement pénalisés. Cette pénalisation peut se traduire par une augmentation importante du temps de réponse du système pour leurs travaux, une impossibilité de mise en oeuvre de certains programmes, ou simplement des sanctions financières. Deux attitudes sont alors possibles pour ces utilisateurs : passer les programmes sur une autre installation ou les modifier afin qu'ils puissent être traités sur l'installation de manière plus satisfaisante.

Nous avons envisagé l'impact de modifications dans des cas d'utilisation plus restrictive des ressources, mais le phénomène se produit de manière analogue dans le cas contraire. Ainsi une plus grande souplesse ou facilité dans l'utilisation des fichiers-disque à accès direct entraînera sans doute leur plus grande utilisation, si cette modification correspond à un besoin latent. Cela entraînera par suite une plus grande utilisation des canaux-disque, et peut-être une diminution de l'usage des bandes magnétiques etc. Une telle modification va donc jouer sur de nombreux paramètres et de manière délicate à évaluer a priori.

Il est certain que l'impact d'une modification sera d'autant plus difficile à évaluer que la population utilisatrice sera changeante.

Envisageons par exemple une installation destinée à l'origine au calcul scientifique essentiellement, donc sans beaucoup de périphériques.Supposons que le besoin se fasse sentir de traiter des travaux de gestion. Cela va entraîner des modifications au niveau du software et de la charge de l'installation, mais surtout il va falloir envisager un équipement en disques ou en dérouleurs de bandes magnétiques et périphériques lents adaptés à la gestion. Dans ce choix, de très nombreux facteurs interviennent (budget prévu, fiabilité

du matériel etc...) et notre modèle pourra servir d'aide à la décision. Pour cela il faudra, a priori, estimer l'impact des modifications envisagées sur les paramètres et appliquer dans les différentes alternatives le modèle. Cette première application pourra suggérer dans les différents cas des améliorations (exemples : couplage, gestion de l'entrée des travaux en mémoire, choix des éléments du moniteur résidents en mémoire centrale etc...). Lorsqu'on aura déterminé un optimum pour chaque alternative on aura constitué un élément important d'aide à la décision. Bien entendu ce genre de travail nécessite une connaissance très approfondie de l'installation et de la population des utilisateurs.

L'adaptation progressive des utilisateurs lors de l'implantation de nouvelles possibilités sur une installation nous fait donc voir la difficulté d'évaluer a priori l'impact d'une modification. Cette constatation peut d'ailleurs se faire dans tous les domaines où intervient le comportement humain, puisque, comme nous le verrons, les paramètres qu'il faut évaluer font, pour la plupart, intervenir les caractéristiques des travaux traités, c'est-à-dire le comportement des utilisateurs.

Par ce biais nous avons rejoint les problèmes d'échantillons de programmes représentatifs de la charge d'une installation (benchmarks), utilisés en particulier lors du choix d'une machine (choix du type et choix des éléments du hardware et du software nécessaires).

Après avoir vu toute la part d'hypothèse qui réside dans l'évaluation a priori des paramètres, nous allons nous intéresser au cas concret de l'évaluation a postériori des paramètres, c'est-à-dire l'évaluation des paramètres sur une installation qui tourne, en mettant en évidence les éléments qui les influencent.

V.3. Evaluation "a postériori" des paramètres :

Les paramètres dont la connaissance est nécessaire

à la mise en oeuvre du modèle vont demander différents modes d'évaluation, selon ce qu'ils mesurent.

Nous verrons en détail ce qui nous semble convenir à chacun, mais nous allons auparavant développer un certain nombre de points qui sont communs à plusieurs d'entre eux.

V.3.1. Les techniques de mesure utilisables :

Nous avons vu, au chapitre I, que, parmi les mesures, certaines étaient locales, c'est-à-dire s'intéressaient au comportement d'un point particulier du système, alors que d'autres, globales, essayaient d'appréhender le comportement du système dans son ensemble.

Cette distinction se retrouve ici. En effet les paramètres que nous étudions peuvent faire l'objet d'études locales : ce sera par exemple l'étude du temps que mettra un disque à têtes mobiles pour satisfaire une demande de service, en fonction du nombre de demandes en attente et/ou de la politique de gestion du disque. Des études locales sont même dans certains cas nécessaires à une bonne connaissance du phénomène étudié et donc à l'évaluation précise de certains paramètres : ainsi l'étude des demandes de pages lorsqu'on utilise le chargement en mémoire des pages "à la demande".

Mais en général les mesures que nous proposons entrent dans le cadre de mesures globales, c'est-à-dire qu'un même procédé va nous permettre de suivre le comportement de l'ensemble du système et d'évaluer les paramètres qui s'y rapportent.

es mesures globales peuvent être câblées ou programmées, et dans ces dernières nous avons vu deux types : les mesures par échantillonnage et les mesures de même type que la comptabilité.

Nous ne reviendrons pas sur l'intérêt des mesures câblées (cf. I.2.1.), dont la mise en oeuvre, toutefois, peut nécessiter un gros investissement, surtout si le nombre de points de mesure est élevé.

Restent les mesures programmées. Etant donné le type des paramètres que nous avons à évaluer (probabilité pour un évènement d'arriver entre t et t + dt) la méthode par échantillonnage nous semble peu adaptée et certainement correspondra beaucoup moins bien à nos désirs qu'une méthode d'évaluation basée sur la comptabilité (cf. I.2.2.2.2.).

Aussi proposerons-nous souvent l'utilisation des résultats de la comptabilité de l'exploitation. Les avantages en sont l'obtention, sans perturbation du fonctionnement de l'installation, de statistiques portant sur la charge et le comportement réel de l'installation pendant le temps que l'on voudra, et pour un coût minimum. Les inconvénients sont, bien sûr, que la comptabilité donne le plus souvent des résultats très globaux, avec des unités de temps trop grandes, et que certains renseignements qui nous sont nécessaires n'y figurent pas. L'utilisation de la comptabilité nécessite une connaissance approfondie du système afin de déterminer exactement ce qui est mesuré et comment cela est effectué.

A titre d'exemple nous pouvons citer quelques renseignements donnés par la comptabilité, par travail ou par étape (une étape d'un travail est une partie du temps de traitement de ce travail pendant laquelle un processeur est activé) : on y trouvera le temps d'utilisation de l'unité centrale, la place prise en mémoire centrale et sur disques rapides, le nombre d'octets transférés entre la mémoire et les périphériques des différents types, le nombre de cartes lues ou perforées etc ...

Mais, lorsque sur un point les renseignements donnés par la comptabilité sont insuffisants ou inexistants, ou bien si on veut obtenir des résultats plus détaillés, comme(par exemple un histogramme reflétant la fréquence d'arrivée de tel évènement, alors que la comptabilité ne donne que son nombre d'arrivées pendant un intervalle de temps qui peut être assez long),alors la comptabilité ne suffit pas.

Dans ce cas nous proposons de mettre au point, ou d'utiliser si elle existe sur le système, une méthode dont la manière de procéder est analogue à celle utilisée par la comptabilité.

Cette méthode est basée sur l'utilisation de pièges et pourra être utilisée pour évaluer les paramètres ζ_i, β, χ_n, λ, ω_i, ϖ_i, μ, ω, σ, σ', γ_i etc... et de manière plus générale tous les paramètres se rapportant à des événements dont l'apparition nécessite le branchement à un module de traitement particulier.

La méthode consiste à piéger ces modules, c'est-à-dire à créer lors de leur exécution un déroutement vers une séquence d'instructions destinée à l'évaluation ; le déroutement ne s'effectuera que lorsque l'évaluation sera lancée.

Les instructions supplémentaires seront plus ou moins nombreuses selon que les renseignements désirés seront plus ou moins nombreux et détaillés, mais aussi selon l'allongement du temps de traitement de l'événement que l'on peut admettre, et qui pourra varier selon le type d'événement et selon l'espace mémoire dont on pourra disposer.

On pourra par exemple incrémenter simplement un compteur chaque fois que le module signalant l'arrivée d'un certain événement sera traité : on saura ainsi combien de fois cet événement s'est produit pendant tout le temps qu'a duré la mise en oeuvre du dispositif d'évaluation. Ou bien on pourra, lors du traitement de l'événement, consulter une horloge du système qui permettra de le dater ; si on a gardé en mémoire l'heure à laquelle le dernier événement de ce type s'est produit, on pourra déduire la durée de l'intervalle de temps séparant deux occurences d'événements de même type. Ce renseignement pourra être alors entré directement dans une table qui servira à étudier la loi d'arrivée de l'événement.

Il faut noter qu'une telle méthode répertorie un événement lorsqu'il est pris en considération. Le décalage entre son arrivée et son traitement pourra donc être important. Ceci ne doit pas être négligé si on étudie les lois d'arrivées d'événements dans le temps.

La perturbation entraînée par l'implantation d'un tel dispositif dépendra du nombre de paramètres à évaluer, de la fréquence d'arrivée des événements sur lesquels portent la mesure et enfin, du temps de calcul et de la place en mémoire centrale et secondaire nécessaires à la mesure.

Il semble qu'on puisse rendre cette perturbation peu importante, d'autant plus qu'il est possible de ne pas évaluer tous les paramètres simultanément, mais au contraire, sur la base d'une étude statistique de la charge de l'installation, établie par exemple à l'aide de la comptabilité, d'évaluer différents paramètres à différents moments, ce qui rend alors la perturbation minimale.

L'évaluation des paramètres sera lancée lors du passage d'un échantillon de programmes représentatif de la charge de l'installation, ou bien on déterminera à l'aide de statistiques les heures et les durées de mise en oeuvre du processus.

Ce procédé de mesure que nous proposons peut ne pas convenir à l'évaluation de certains paramètres pour lesquels, nous l'avons vu, une étude particulière semble nécessaire. On utilisera alors des mesures câblées ou programmées, en ayant soin là aussi d'évaluer, pour ces dernières, les perturbations qu'elles entraînent sur ce qu'elles mesurent. Nous verrons cela en détail pour les différents paramètres concernés.

V.3.2. Etude particulière de chaque paramètre du modèle.

Après avoir vu de manière générale comment procéder pour obtenir la valeur des paramètres qui nous sont nécessaires, nous allons reprendre séparément l'étude des différents paramètres.

Pour cela nous les avons répartis en trois types que nous étudierons successivement :

. les paramètres décrivant des caractéristiques essentiellement hardware de l'installation.

. les paramètres qui dépendent principalement de la population des utilisateurs.

. enfin les paramètres dans lesquels le software propre à l'installation joue un rôle primordial, en liaison le plus souvent avec les caractéristiques de la population des utilisateurs.

V.3.2.1. Les paramètres liés au hardware.

Dans cette catégorie nous trouvons :

θ_1 : le nombre d'unités de temps pour faire un tour de disque rapide.

θ_2 : le nombre d'unités de temps pour faire un tour de disque lent.

k_1 : le nombre de secteurs d'un disque rapide.

k_2 : le nombre de secteurs d'un disque lent.

a : le nombre de ressources de types différents, essentiellement organes périphériques, accessible aux utilisateurs.

b_i : $i = 1...a$, le nombre d'éléments de la ressource i

γ et les γ_i : $\gamma_i dt$ est la probabilité pour qu'entre t et t + dt le périphérique lent n°i demande accès à une page-disque rapide (d'un fichier-symbiont).

ζ_i : $\zeta_i dt$ est la probabilité pour qu'un organe de type i, lorsqu'il est en service, termine une entrée-sortie entre t et t + dt.

k'_2 : nombre de secteurs d'un disque lent servis en un tour, en moyenne, lorsque la file d'attente de service du disque n'est pas vide.

V.3.2.1.1. $\theta_1, \theta_2, k_1, k_2, a, b_i$.

Ces six paramètres ne posent pas de problème d'estimation puisqu'on les obtient directement au vu de l'installation et de la documentation qui s'y rapporte.

V.3.2.1.2. γ_i.

La connaissance de γ_i va, elle, faire intervenir en plus le software de base des différents périphériques lents, leur taux d'utilisation, et aussi le comportement des utilisateurs dans certains cas.

Il n'y aura pas de difficulté à établir γ_i lorsque chaque caractère lu ou écrit sur un périphérique lent a son image sur le disque, c'est-à-dire lorsqu'on n'utilise pas de compression (remplacement de successions de caractères identiques par une séquence comprenant un signe distinctif, le caractère et le nombre de ses répétitions). La connaissance de la cadence de transmission du périphérique lent suffira alors. En revanche s'il y a compression il faudra estimer le taux de compression des informations transmises (cf V.3.2.2.1.).

Nous pouvons donner un exemple.

Supposons qu'en cours d'exploitation, l'imprimante fonctionne a % du temps.

Elle peut écrire n caractères par ligne à raison de b lignes par seconde en moyenne. Il n'y a pas de compression.

Un caractère est stocké sur un octet, mais pour chaque ligne on a de plus transmission de c caractères de contrôle de sorte que chaque ligne correspondra à (n+c) = d octets.

Une page disque contenant e octets, chaque page disque contiendra donc l'information nécessaire à l'écriture de e/d = f lignes.

L'imprimante demandera donc une page de disque toutes les f/b secondes, soit toutes les g unités de temps.

Ce qui, compte tenu du taux d'activité de l'imprimante nous donne,

$$\gamma_1 = \frac{a}{100g} \quad \text{ou} \quad \gamma_1 = \frac{ab(n+c)}{100e}$$

Si on a un taux de compression de h portant sur les informations transmises et non sur les caractères de contrôle, on aura alors

$$\gamma_1 = \frac{ab(nh+c)}{100e}$$

V.3.2.1.3. ζ_i.

Les paramètres ζ_i, qui caractérisent le temps de réponse des périphériques autres que les disques n'utilisant pas de symbiont, sont bien sûr avant tout des paramètres de hardware, mais ils sont aussi liés au software, et/ou aux utilisateurs, selon les cas.

Les caractéristiques hardware des différents organes jouent bien sûr un rôle primordial sur ζ_i, mais leur répartition entre les différentes unités de liaison ne saurait être négligée (cf II.2.3).

L'influence du software et aussi des utilisateurs se fera sentir, pour les unités de bandes magnétiques, par l'organisation des fichiers et les types de fichiers utilisés.

Pour les consoles utilisateurs, le temps de réponse sera lié tout à la fois aux performances propres à la console (ex : présence ou non de lecteur - perforateur de ruban ou de cartes, rapidité d'affichage des messages etc) et au délai de réaction, ou "temps de réflexion", de l'utilisateur à sa console.

L'estimation des différents ζ_i est donc plus délicate que celle des paramètres précédents.

Plusieurs possibilités sont envisageables, d'une part nous l'avons vu, la méthode développée au paragraphe V.3.1. ou bien une méthode de mesures câblées qui fait des tests sur

l'activité des différents canaux. Dans ce dernier cas, il faudra détecter l'arrivée, pour chaque organe, d'une demande d'entrée/sortie et la fin de son exécution et consulter pour chacune une horloge afin de déterminer le temps écoulé, qui sera donc le temps de réponse de l'organe.

On peut aussi envisager d'utiliser la comptabilité : on y trouve pour chaque travail et par type de ressource, le nombre d'appels au superviseur d'entrée-sortie et le nombre de caractères transférés.

Connaissant la cadence de transfert de la ressource considérée, on pourra donc déduire le temps de transfert moyen, en faisant intervenir, s'ils n'ont pas encore été pris en considération, les caractères de contrôle transférés.

Nous connaitrons ainsi, en moyenne, le temps de transfert par type de ressource .

Pour obtenir le temps de réponse, il reste à estimer le temps d'accès à l'information qui variera selon le type d'organe et l'utilisation que l'on en fait.

Ainsi pour des bandes magnétiques utilisées comme support de fichiers séquentiels, le temps d'accès à l'information sera simplement le temps de passage sur l'intervalle séparant deux blocs d'information, sauf dans les cas de recherche de début et de fin de fichier, et on peut estimer la probabilité d'une telle recherche.

Etudions pour les consoles utilisateurs le temps écoulé entre le lancement d'une demande d'entrée-sortie et la fin de son exécution.

Deux cas peuvent se produire selon qu'il y a ou non intervention manuelle.

S'il y a intervention manuelle nous aurons alors successivement lancement d'un message de la machine à l'utilisateur, puis temps de "réflexion" ou de "réaction" de l'utilisateur, suivi de sa réponse. Il faut donc étudier les temps de transfert des messages et le temps de réaction de

l'utilisateur.

Au contraire, dans le cas où il n'y a pas d'intervention manuelle, comme par exemple s'il faut lancer une lecture au lecteur de ruban de la console, il s'agit d'un cas habituel d'entrée-sortie avec temps d'accès à l'information et temps de transfert. On peut donc le traiter comme nous l'avons indiqué plus haut.

V.3.2.1.4.k'_2 :

Le paramètre k'_2 indique le nombre de secteurs d'un disque lent lus ou écrits, en moyenne, en un tour de disque, lorsque la file des demandes de transfert du disque n'est pas vide.

Ce paramètre fait intervenir la politique de gestion des entrées/sorties du disque dont nous avons vu une étude en I.3.1.3.

En effet, pour les disques lents , le nombre de secteurs lus ou écrits en un tour du disque dépendra de la méthode employée pour le choix des mouvements de bras.

Cette méthode sera choisie en fonction de l'usage que l'on attend du disque. Ainsi, si les demandes sont assez peu nombreuses, par exemple en moyenne 1 pour n tours de disque, on pourra choisir simplement de traiter les demandes dans leur ordre d'arrivée. Au contraire si les demandes sont nombreuses et risquent d'entraîner de longues files d'attente, il sera nécessaire d'envisager une méthode de gestion du disque plus performante pour une telle charge. On pourra par exemple utiliser le positionnement des têtes de lecture-écriture successivement sur chaque cylindre pendant N tours et donc servir les demandes en attente dans l'ordre des cylindres.

Par suite, pour certaines politiques de gestion, le nombre de secteurs servis en un tour pourra être lié à la longueur d de la file d'attente D de service du disque à ce moment là.

La technique employée pour évaluer k'_2 dépendra donc de l'algorithme utilisé pour servir les demandes en attente: dans certains cas les résultats de mesures câblées ou du type développé en V.3.1. suffiront, alors que pour d'autres il sera nécessaire, pour bien étudier le phénomène, de développer un modèle analytique ou une simulation (cf à ce sujet Teorey et Pinkerton - Réf [48]) du comportement du disque.

Il faut noter que la comptabilité est insuffisante ici pour la mise en évidence de k'_2, car les renseignements qu'elle nous donne ne nous permettent pas de mettre en évidence le temps d'accès à l'information, alors que le temps de transfert est connu : c'est le temps θ_2/k_2 mis par un secteur du disque pour défiler devant les têtes de lecture.

V.3.2.2. Les paramètres liés principalement à la population des utilisateurs:

Nous avons regroupé dans cette catégorie trois paramètres :

ν : le nombre de pages disque demandées par un travail utilisateur en entrée depuis un lecteur de cartes est $1/\nu$

β : β dt est la probabilité qu'un travail opérationnel se termine entre t et t + dt.

χ_n : χ_n dt est la probabilité qu'une $n^{ième}$ console utilisateur soit mise en service entre t et t + dt.

Nous allons les étudier successivement.

V.3.2.2.1. ν :

Ce paramètre dépend directement du nombre de cartes perforées lues par travail puisque ce sont ces dernières qui sont transcrites sur le disque (cf pour cela le paramètre γ_i § V.3.2.1.2.) et aussi du taux de compression que vont subir les informations entre leur lecture sur cartes et leur rangement en fichier sur disque.

Le nombre de cartes par travail dépend essentiellement du type de travaux traités sur l'installation : essais ou travaux de routine, travaux scientifiques ou de gestion etc.

Mais le matériel intervient aussi : si le lecteur est peu performant mais que l'installation dispose de ressources intéressantes en disques, bandes magnétiques, etc, il est probable que les utilisateurs préféreront, dans la mesure du possible, se servir de ces dernières en y stockant par exemple des modules de programmes ou des données. Ceci sera d'autant plus vrai que l'utilisation de ces ressources sera rendue aisée et puissante par le software proposé. (Ex catalogage des cartes de commandes etc).

Il faut noter qu'un travail entré par lecteur de cartes comprend en général des cartes-programmes et des cartes-données. Il sera considéré comme présent dans la file E à partir d'un certain moment déterminé par le régulateur et qui pourra être, par exemple, dès que les cartes contenant les informations nécessaires au lancement de sa première étape seront entrées. Mais le travail suivant sur le lecteur ne pourra être entré dans la file E que lorsque toutes les cartes du premier auront été entrées. Par suite, les entrées de deux travaux successifs dans la file E seront bien séparées par la durée moyenne d'entrée d'un travail sur le lecteur de cartes, si celui-ci travaille sans arrêt. Si pendant la période où l'installation travaille en régime permanent, le lecteur n'a un taux d'utilisation que de $y < 1$, il faudra bien sûr faire intervenir ce taux pour évaluer le temps moyen séparant l'entrée de deux travaux dans la file E : la probabilité d'avoir entre t et t + dt une nouvelle entrée dans la file E sera non plus $\nu \gamma_1$ dt mais $\nu y \gamma_1$ dt.

On pourra perfectionner le modèle encore en utilisant $\nu \gamma_1$ dt. (Probabilité pour qu'à l'instant t le lecteur travaille)
cette dernière probabilité pouvant dépendre du nombre de travaux présents dans la file E, ou de la place occupée par ces travaux dans les fichiers-disque du système, (ceci est une

méthode destinée à éviter la saturation des disques), ou encore d'autres facteurs.

L'estimation du paramètre ν pourra se faire en utilisant la méthode exposée en V.3.1. et fonctionnant de manière analogue à la comptabilité. Il faudra alors mettre un piège dans le symbiont correspondant au lecteur de cartes afin de repérer la fin de l'entrée d'un programme : on ira alors rechercher dans les tables du système, à l'aide d'un module de mesure, la taille en pages-disque du programme entré.

Une autre manière de faire consistera à déduire ν du nombre de cartes lues par programme, ce renseignement figurant dans la comptabilité. La déduction sera immédiate s'il n'y a pas de compression. Si au contraire il y a compression il faudra estimer le taux de compression qui est le rapport du nombre de caractères lus par carte au nombre de caractères rangés sur disque. On pourra obtenir le taux de compression au moyen d'un programme qui analysera un échantillon de cartes représentatif de l'ensemble des cartes lues au lecteur.

V.3.2.2.2. β :

La probabilité qu'un travail se termine dépend bien sûr du type de travaux passés sur l'installation (scientifiques, gestion, routine, essais).

Ici nous voyons apparaître le rôle important de la détection des erreurs permettant la mise au point d'essais en un petit nombre de passages, si elle est bien faite.

La fiabilité du matériel apparaîtra elle aussi à ce niveau, surtout en ce qui concerne les périphériques. Elle aura une incidence particulièrement importante lorsque l'on traitera de gros travaux, surtout de gestion, utilisant de nombreux fichiers.

La comptabilité de l'exploitation donne pour chaque travail le temps pendant lequel il a utilisé l'unité centrale :

c'est précisement ce renseignement qui nous intéresse et on en déduira β : ce sera l'inverse du temps moyen d'utilisation de l'unité centrale.

V.3.2.2.3. χ_n :

La probabilité d'avoir une mise en service d'une $n^{ième}$ console va dépendre de nombreux éléments liés au comportement des utilisateurs. A priori on peut penser que cette probabilité variera selon les heures de la journée, les périodes de l'année, du mois ou de la semaine. (cf. J.Maublanc Réf 62).

Le nombre de consoles auxquelles pourront avoir accès les utilisateurs aura une incidence, ainsi que le comportement du système au fur et à mesure de l'accroissement de sa charge en consoles en service, en particulier l'allongement du temps de réponse qui en résulte.

Pour estimer χ_n on pourra bien sûr utiliser la méthode développée en V.3.1. Pour cela on créera un tableau contenant autant de mots plus un qu'il y a de consoles sur l'installation, soit $b_3 + 1$. Chaque fois qu'arrivera une interruption signalant la mise en/hors service d'une console on ira consulter une horloge du système. On calculera le temps écoulé depuis la dernière interruption de ce type, dont on aura préalablement gardé l'heure d'arrivée. On pourra donc en déduire l'intervalle de temps pendant lequel on a eu n consoles en service. La durée de cet intervalle de temps sera ajoutée au contenu de la case numéro n+1 du tableau. A la fin de l'évaluation, on saura pendant combien de temps on a eu 0, 1 ..... b_3 consoles en service, ce qui nous permettra de déduire χ_n.

Si on n'a pas homogénéité dans le temps, par exemple s'il existe des fluctuations journalières, on pourra déterminer les paramètres χ_n correspondant à chaque période : par exemple les χ_n $n = o \ldots b_3$ du matin et ceux de l'après-midi.

On peut aussi envisager d'obtenir χ_n à partir des résultats donnés par la comptabilité de l'exploitation à condition que celle-ci donne non seulement la durée globale des différentes sessions mais aussi les heures de début et de fin de session pour chaque console, à moins qu'elle n'établisse dans ses statistiques les pourcentages de temps pendant lesquels il y a eu n consoles en service pour $n = o \ldots b_3$, ce qui nous conviendrait mieux encore.

V.3.2.3. Paramètres liés essentiellement software :

Dans cette partie nous allons étudier les paramètres qui dépendent essentiellement du software, mais nous verrons que pour la plupart d'entre eux l'incidence des utilisateurs ne saurait être négligée.

Nous verrons successivement :

λ : λdt est la probabilité d'un réveil du régulateur entre t et t + dt.

ω_i : ω_i dt est la probabilité de lancement d'une demande d'entrée/sortie sur un organe de type i, entre t et t + dt pour une tâche utilisateur.

α_i : probabilité qu'une demande d'entrée-sortie sortie sur un organe du type i entraîne le passage à l'état "en attente d'entrée-sortie" de la tâche qui l'a lancée.

σ et σ': probabilité que le moniteur travaille pour lui-même lorsqu'il y a au moins une tâche active, respectivement il n'y a pas de tâche active, d'un utilisateur.

ω_o : ω_o dt est la probabilité de demande d'entrée/sortie disque rapide par le moniteur lorsqu'il travaille pour lui-même, entre t et t + dt.

μ_i : μ_i dt probabilité d'avoir entre t et t + dt une mise en attente d'entrée-sortie d'une tâche sans lancement simultané d'une entrée-sortie, la tâche ne pouvant

redevenir active qu'à la fin d'une entrée-sortie sur un organe de type i.

$\tau_{n+1}(b_i)$: probabilité pour que n+1 travaux, les n présents dans la file V et le candidat au lancement demandent au plus b_i éléments de la classe de ressource i sachant que n travaux, les n travaux de la file V, demandent au plus b_i.

V.3.2.3.1. λ :

Nous avons vu que différentes causes peuvent lancer l'activation du régulateur (cf II.3.2.). Cet événement arrive avec une probabilité λ dt entre t et t+dt, λ pouvant être fonction de n le nombre de travaux présents dans V.

L'estimation de λ peut se faire à l'aide de la méthode exposée en V.3.1. en plaçant un piège dans le régulateur. Le piège entraîne la mise en oeuvre d'un module de mesure à chaque réveil du module du régulateur chargé de la consultation de la file E en vue du lancement de travaux.

Si l'on veut simplement estimer λ, il faudra incrémenter un compteur à chaque réveil de ce module du régulateur. Lorsqu'on arrêtera l'évaluation on aura le nombre de consultations de la file E pour lancement de travaux effectuées par le régulateur pendant la durée de l'évaluation : on en déduira immédiatement λ.

Si de plus on veut voir la variation de en fonction de n il faudra à chaque réveil du module de mesure tester le nombre de travaux présents dans la file V, en consultant les tables du système. On incrémentera le n+1ième compteur d'un tableau ayant autant d'entrées plus une qu'il peut exister de travaux coexistant dans la file V. De là on pourra déduire λ et voir si l'on peut accepter l'hypothèse qu'il dépend de n.

La comptabilité ne donne pas de renseignement sur le comportement du régulateur, en général.

V.3.2.3.2. ω_i :

Les paramètres ω_i sont des paramètres particulièrement importants dans le système, comme le montre notre modèle, puisqu'ils étudient les occurences de demandes d'entrée/sortie.

Nous devons distinguer deux types d'entrées-sorties que nous étudierons successivement selon qu'elles sont nécessaires au déroulement du travail ou qu'elles ont été introduites pour les besoins du système.

V.3.2.3.2.1. Les entrées-sorties nécessitées par le travail :

Une distinction s'impose entre :

. entrée-sortie "réelle" qui est une entrée-sortie réalisée physiquement par un organe périphérique

. et entrée-sortie "virtuelle" qui est un appel demandant au moniteur d'exécuter les opérations nécessaires à l'entrée (resp. à la sortie) d'informations depuis (resp. vers) un organe périphérique vers (resp. depuis) une zone désignée du travail demandeur.

Lorsqu'un travail lance une entrée virtuelle, la situation sera différente selon que le moniteur pouvait ou non "s'attendre" à la demande d'entrée de cette information précise. Ainsi, lors du traitement en entrée d'un fichier séquentiel le processus suivant se déroule : lecture d'un enregistrement, son traitement, lecture de l'enregistrement suivant etc ... Lorsqu'un enregistrement est demandé, le suivant aura donc toutes chances d'être demandé à son tour, sauf si le processus est interrompu par la volonté du programmeur ou à la suite d'une erreur ou d'une anomalie.

Au contraire, lors de l'ouverture d'un fichier, ou de la recherche d'un enregistrement dans un fichier à clé., on ne peut pas savoir a priori quel enregistrement va être demandé.

S'il n'est pas possible de prévoir quel enregistrement va être demandé l'entrée réelle ne pourra être lancée que lorsque l'entrée virtuelle correspondante sera faite. De plus, à une entrée virtuelle correspondra au moins une entrée réelle : il y

en aura en effet plusieurs s'il n'est pas possible d'accéder directement à l'enregistrement (exemples chaînages, fichiers à clé...).

En revanche si la prévision est possible, l'entrée réelle va pouvoir être lancée avant l'entrée virtuelle correspondante et le résultat pourra être stocké dans une zone tampon du système jusqu'à ce que l'entrée virtuelle soit lancée. Il n'y aura plus alors qu'à transférer l'information depuis une mémoire-tampon du système vers la zone utilisateur.

Bien entendu s'il y a suffisamment de place dans les zones tampons, on pourra y stocker non un mais plusieurs enregistrements à l'avance. Cela est particulièrement intéressant lorsque la lecture se fait par blocs contenant plusieurs enregistrements : dans ce cas, on aura donc une seule entrée réelle pour plusieurs entrées virtuelles.

La situation est analogue pour les sorties.

Ce sont bien sûr les entrées-sorties réelles qui nous intéressent, mais nous voyons que leur fréquence est régie non seulement par la fréquence des entrées-sorties virtuelles, donc si on veut, le type de travail , mais aussi par la taille des enregistrements à lire ou écrire , par la taille et le nombre des mémoires tampons, par les types de fichiers utilisés et par la gestion des fichiers sur l'installation.

Plusieurs méthodes d'évaluation des ω_i sont possibles, selon l'utilisation que l'on veut faire du modèle.

La première méthode nécessite peu d'investissement puisqu'elle utilise la comptabilité, dans la mesure où celle-ci donne, par travail ou par tâche, le nombre d'entrées-sorties réelles par organe ou par type d'organe et le temps d'utilisation de l'unité centrale. Les ω_i se déduisent alors immédiatement puisqu'il suffit de faire le rapport du nombre d'entrées-sorties réelles sur le temps d'utilisation de l'unité centrale. Ceci bien sûr en prenant un temps d'utilisation suffisamment long et un échantillon représentatif pour que le résultat

soit statistiquement valide.

Si la comptabilité ne donne pas ces renseignements, il faudra les obtenir directement par mesures (par câblage sur l'utilisation des canaux ou bien par la méthode exposée en V.3.1). Il est même possible de calculer les ω_i correspondant différents types de tâches : compilation, édition de liens, exécution d'un travail de gestion ou d'un travail scientifique etc...

On en déduira les ω_i de l'ensemble en pondérant les différents ω_i partiels par les fréquences relatives établies à partir des temps de calcul des différents types de tâches et en en faisant la somme.

Ceci peut permettre d'établir plus aisément l'incidence de variations de la charge de l'installation selon les différents types de tâches : il suffira de faire varier les facteurs de pondération utilisés.

On peut enfin envisager de déduire les ω_i d'une connaissance d'une part des entrées-sorties virtuelles et d'autre part du traitement auquel elles donnent lieu selon les cas (type d'organe, taille des enregistrements demandés, type du fichier etc...).Cette approche est la plus délicate et pourra nécessiter l'utilisation d'un modèle ou d'une simulation locale, mais par la connaissance approfondie du système qu'elle procure, c'est la seule qui permette d'étudier a priori les répercussions de certaines modifications dans l'utilisation des fichiers, des mémoires_tampons (nombre, taille) etc...

De toute façon la comparaison des nombres d'entrées-sorties réelles et virtuelles est très intéressante. Et si l'on considère les entrées-sorties virtuelles comme une donnée, on peut envisager une optimisation "locale" du système qui diminuerait si possible le nombre d'entrées-sorties réelles sans pour cela demander "trop" de place en mémoire centrale.

Mais il ne faut pas perdre de vue, les développements précédents l'ont montré, que dans un système d'exploitation l'interaction très poussée des différents éléments nécessite, même lors d'une modification locale, une évaluation globale du comportement de l'installation.

Ainsi une diminution notable du nombre d'entrées-sorties réelles, obtenue en augmentant, par exemple, la taille ou le nombre des mémoires_tampons peut être ou non favorable selon les répercussions qu'elle a sur le reste du système, exemple : demande de place supplémentaire en mémoire centrale.Ceci est vrai sauf bien sûr dans le cas évident où une modification consisterait simplement en une amélioration locale sans contrepartie notable : ce serait le cas, toutes choses étant égales par ailleurs, d'une amélioration d'un processeur le rendant plus rapide et cela sans contrepartie ; nous sommes là aux confins des erreurs de performances.

Nous nous sommes intéressés jusqu'alors aux entrées-sorties réelles, issues d'entrées-sorties virtuelles lancées au fur et à mesure du traitement d'un travail.

Ces entrées-sorties sont des entrées-sorties d'éléments de fichiers nécessaires au bon déroulement du travail. Ceci englobe, rappelons-le, tout aussi bien l'entrée en mémoire d'un processeur que l'écriture des résultats d'un calcul dans un fichier de sortie. Globalement on peut dire que ces entrées-sorties sont nécessaires à la mise en oeuvre du travail utilisateur.

V.3.2.3.2.2. Les entrées-sorties disques rapides liées à la gestion des travaux en mémoire centrale.

Un second type d'échanges existe, mais seulement entre la mémoire centrale et les disques rapides et ne concerne donc que le paramètre ω_1.

Il s'agit d'entrées - sorties liées directement au fonctionnement du système : c'est le "swapping", qui consiste

en l'entrée et la sortie de la mémoire centrale de pages-utilisateurs en fonction d'un algorithme, et ceci dans un but d'optimisation, selon certains critères de l'exploitation. Il faut bien distinguer ces entrées-sorties des entrées-sorties d'éléments du moniteur destinés à son usage propre et étudiées par le paramètre ω_1.

Le paramètre ω_1 cherché doit bien sûr faire intervenir les deux types d'entrées-sorties et l'évaluation de la contribution du second va poser des problèmes puisque, contrairement au premier, lié essentiellement au travail, il fait intervenir de manière primordiale le contexte dans lequel se fait la mise en oeuvre du travail.

D'un cas à l'autre, la situation change et les problèmes posés par l'évaluation sont différents.

Certains cas sont relativement simples à traiter comme celui où chaque travail candidat, successivement, est entré en mémoire centrale, s'exécute pendant un certain quantum de temps puis est ressorti afin de laisser la place à un autre. Le temps de swapping est alors utilisé par un autre travail déjà présent en mémoire. Dans un cas comme celui ci il semble que des mesures faites directement sur l'installation en cours de fonctionnement (comptabilité, câblage ou méthode exposée en V.3.1.) soient suffisantes puisque les résultats dépendent essentiellement des caractéristiques individuelles des travaux.

En revanche, dans d'autres cas de swapping, comme le chargement des pages à la demande, l'environnement dans lequel se trouve chaque travail va avoir un rôle essentiel.

Ici par exemple, un travail va se voir attribuer un certain quantum de temps pendant lequel il va pouvoir s'exécuter. Le travail ne sera pas entré en totalité en mémoire centrale, mais, au fur et à mesure de ses demandes on entrera les pages nécessaires à son bon déroulement. Ceci permet de n'entrer en mémoire que les pages réellement nécessaires à la mise en oeuvre du travail et évite donc des échanges de page inutiles

entre les disques et la mémoire. Par voie de conséquence, cette méthode peut permettre d'augmenter le taux de multi-programmation (nombre de travaux présents en mémoire, au moins en partie). Chaque page entrée écrase bien sûr une page présente antérieurement en mémoire centrale, et qui était allouée, selon les cas et l'algorithme de remplacement choisi, à l'utilisateur dont le quantum de temps s'écoule actuellement, ou à un autre.

A la fin du quantum de temps le contrôle est remis à un autre travail. S'il a été présent en mémoire antérieurement certaines pages qu'il y avait auront pû être écrasées par l'entrée de pages destinées à d'autres travaux, selon l'algorithme de remplacement et selon les caractéristiques propres à ceux-ci.

En effet on remarque (cf Smith, Réf. [45], Denning Réf. [11]) que pour s'exécuter correctement un travail a besoin, à un certain moment, d'un certain nombre de pages. Ce nombre pourra varier au fur et à mesure du déroulement du travail. S'il n'a pas à sa disposition cet espace de travail (working-set) il passera son temps à demander des entrées-sorties de pages, au point qu'on parlera "d'explosion des demandes de page".

La taille de l'espace de travail variera avec le type de travail, mais aussi avec les caractéristiques des modules à exécuter créés sur l'installation étudiée, c'est-à-dire la disposition des différentes zones (instructions, données, réservations etc ...) à l'intérieur d'un module. Ainsi un appel moniteur qui doit se référer à trois zones pourra ne demander aucune page si toutes ces zones sont dans la même page que lui, ou demander par exemple six pages si chaque zone est située à cheval sur deux pages différentes de celles où sont situées les autres, et non encore présentes en mémoire centrale.

Par suite si l'espace de travail est plus grand que la zone allouée au travail, il devra entrer et sortir sans arrêt les mêmes pages, et donc surcharger les canaux sans pour autant

pouvoir donner suffisamment de travail à l'unité centrale. Ceci entraîne un rendement global très mauvais. Ou bien, il devra empiéter sur les zones occupées par d'autres travaux présents en mémoire centrale et un problème analogue risque donc de se poser pour eux dès qu'ils auront le contrôle.

Dans des cas comme celui-ci, l'interaction entre les différents travaux est donc extrêmement importante et pour établir le paramètre ω_1, trois étapes nous semblent nécessaires.

1°) Il faut tout d'abord établir un échantillon représentatif de la charge de l'installation. Ceci peut se faire à partir de la comptabilité qui répertorie tous les travaux traités et donne pour chacun un certain nombre de caractéristiques.

2°) Chaque travail sélectionné dans l'échantillon va être traité séparément de manière à en obtenir une sorte de squelette, au moyen d'un programme de type "Trace".

Voilà ce dont il s'agit : chaque étape d'un travail est représentée matériellement par des informations (instructions, données, réservations etc...). Ces informations sont situées géographiquement dans un espace d'adresses, qui va être découpé en pages numérotées de k mots.

Il va falloir suivre pas à pas l'exécution de chaque étape en repérant :

- pour chaque instruction exécutée dans quelles pages sont situés les éléments auxquels elle se refère : zone opérande, avec éventuellement adressage indirect impliquant de se reporter à des mots pouvant être situés dans des pages différentes etc ...
- et comment se font les passages d'une page à l'autre au fil de la succession en séquence ou par sauts des instructions.

3°) Ces informations, mises en fichier, seront ensuite utilisées dans une simulation ou un modèle local pour déterminer, en fonction de l'algorithme de remplacement et des caractéristiques des travaux, le comportement des travaux admis en mémoire centrale.

Nous pourrons alors, évaluer, en fonction des caractéristiques des travaux de l'échantillon et de leur espace de travail le paramètre ω_1, ou plutôt un paramètre $\omega_1(n)$, n étant le nombre de travaux de la file V. . De plus on pourra être amené par ce biais à discuter la taille des pages et on pourra déduire d'une part des améliorations à apporter aux modules exécutables, afin de diminuer, autant que faire se peut, la taille des espaces de travail et déduire d'autre part l'algorithme optimal de gestion de la mémoire centrale en fonction des caractéristiques de la charge de l'installation et des algorithmes de remplacement utilisables.

Nous aurions pû obtenir ω_1 par des méthodes analogues à celles utilisées au paragraphe V.3.2.3.2.1., mais il semble qu'alors les résultats obtenus n'auraient pas pû être analysés de manière approfondie.

En effet, nous aurions constaté l'entrée ou la sortie de la mémoire de telle ou telle page, mais nous n'aurions pas connu en détail l'évolution des travaux dans la mémoire. Pourquoi telle page est-elle entrée puis sortie cinquante fois de suite de la mémoire ? Est-ce que s'il y avait un travail en moins en mémoire on aurait évité un nombre considérable d'échanges et gagné un temps de calcul important ? Quelles conséquences aurait eu la présence en mémoire d'un travail supplémentaire ?

Autant de questions auxquelles on ne peut répondre sans une analyse très précise et complète du comportement des travaux et cette analyse est nécessaire si l'on veut optimiser ce point très important du système.

V.3.2.3.3. α_i :

Le paramètre α_i étudie, rappelons-le, la probabilité qu'un lancement d'une demande d'entrée-sortie sur un organe de type i entraîne le passage de la tâche qui le demande à l'état "en attente d'entrée-sortie".

Il est le résultat d'évènements de deux types différents selon que la poursuite du travail dépend ou non de la fin d'une entrée/sortie.

V.3.2.3.3.1. Premier type d'évènements:

Rappelons tout d'abord que lors de leurs entrées-sorties les utilisateurs travaillent sur des enregistrements logiques, alors que le système utilise des blocs physiques d'information. La taille d'un enregistrement peut être inférieure, égale ou supérieure à celle d'un bloc.

Dans certains cas un travail peut continuer à s'exécuter et à lancer des demandes d'entrées-sorties, même si les demandes d'entrées-sorties lancées précédemment ne sont pas encore terminées.

Cela arrivera lorsqu'on voudra entrer ou sortir de la mémoire centrale des informations dont la taille est plus grande qu'un bloc, par exemple si on veut entrer en mémoire un module qui occupe trois blocs. On lancera alors plusieurs demandes d'entrée ou de sortie de blocs successivement. Ces demandes sont lancées de manière rapprochée dans le temps et le travail pour le compte duquel le système les lance ne tombera pas en attente d'entrée-sortie avant que toutes ces demandes aient été prises en compte par le superviseur d'entrées-sorties.

Un autre exemple se produit lorsqu'un utilisateur travaille avec plusieurs fichiers simultanément, comme dans le cas de tri ou de fusion de fichiers ou des travaux de gestion en général. La simultanéité de traitement entre l'unité centrale et les périphériques permet alors au travail de lancer, s'il le désire, des entrées-sorties successives sur les différents fichiers sans tomber en attente, ou tout au moins avant de tomber en attente. Il pourra par exemple lancer une sortie sur le fichier F3, puis une entrée sur F1 et une entrée sur F2 ; à la suite de ces trois demandes il tombera "en attente" et ne redeviendra actif que lorsque la première ou les trois auront été satisfaites.

Le paramètre α_1 sera donc déterminé essentiellement, pour ce premier type d'événements, par les caractéristiques propres du travail, en liaison avec les possibilités offertes par le moniteur : ainsi l'existence ou non de swapping sur l'installation aura une influence déterminante sur α_1.

V.3.2.3.3.2. Second type d'événements.

Au contraire, dans le cas où la poursuite ou non du travail dépend de l'instant de fin de transfert, l'environnement dans lequel se déroule le travail va avoir une influence prépondérante.

En effet, dans ce cas, une entrée-sortie est lancée pour le compte d'un utilisateur qui, lui-même, continue à avoir le contrôle de l'unité centrale, c'est le cas lorsqu'une entrée-sortie réelle est lancée avant le lancement de l'entrée sortie virtuelle correspondante. Si l'entrée-sortie se termine avant que l'utilisateur n'ait perdu le contrôle, il va pouvoir continuer à s'exécuter et éventuellement en lancer une autre à nouveau.

Ceci n'est possible en général qu'avec des périphériques suffisamment rapides par rapport à l'unité centrale et une bonne utilisation des mémoires-tampons. L'accès séquentiel et le fait d'avoir plusieurs enregistrements par bloc sont donc favorables. D'autre part, plus le temps de calcul est long entre la demande d'entrée-sortie et sa fin, plus on a de chances d'éviter que le travail ne tombe en attente. Cette situation dans laquelle le résultat d'une entrée-sortie

réelle parvient avant que la tache qui l'a lancée ne soit tombée en attente a donc plus de chance d'arriver lorsque l'on fait du calcul scientifique plutôt que de la gestion et lorsque les files d'attente de service des périphériques sont courtes.

Nous avons vu les causes qui déterminent l'existence de ce paramètre α_i. Nous n'étudierons son évaluation qu'au paragraphe suivant, en même temps que l'évaluation des paramètres μ_i auxquels les α_i sont très liés.

V.3.2.3.4. μ_i.

Rappelons que $\mu_i dt$ est la probabilité d'avoir entre t et t+dt une mise en attente d'entrée-sortie pour une tâche active, sans lancement simultané d'une entrée-sortie, la tâche ne pouvant redevenir active qu'à la suite d'une entrée-sortie sur un organe de type i.

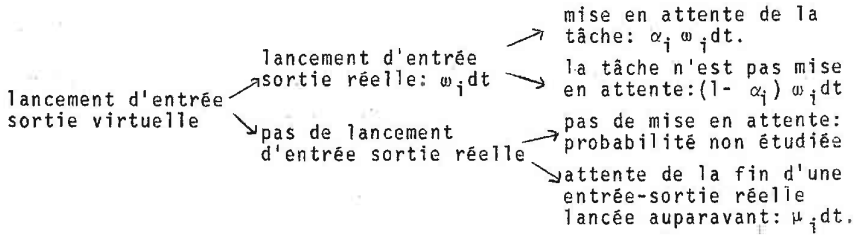
Il est bien clair que nous nous intéressons seulement à une tâche qui passe de l'état "opérationnel" à l'état "en attente", et non au passage de contrôle d'un utilisateur à l'autre, à la suite par exemple d'une suspension par un signal d'horloge d'une tâche qui passe alors d'opérationnelle à prête, même si certaines de ses pages sont sorties de la mémoire pour faire de la place à d'autres travaux.

Une tâche ne peut tomber en attente d'entrée-sortie que de son propre fait. Ceci se produit par une demande d'entrée sortie virtuelle qui, selon les cas, lance une entrée-sortie réelle, probabilité $\alpha_i \omega_i dt$ ou au contraire n'en lance pas, probabilité $\mu_i dt$. Mais dans les deux cas, le travail est dans l'impossibilité de continuer à s'exécuter.

Le passage "en attente d'entrée-sortie" sans lancement simultané d'une demande d'entrée-sortie sera dû au fait, pour une entrée, que l'entrée réelle a déjà été lancée antérieurement par les soins du système, mais n'est pas encore terminée, et pour une sortie, que la zone tampon du système à partir de laquelle doit se faire la sortie réelle est encore occupée par les éléments de la sortie réelle précé-

dente, encore inachevée. Il y a donc eu dans les deux cas lancement antérieurement d'une entrée-sortie qui n'est pas encore achevée.

Nous pouvons résumer les différentes possibilités, avec leur probabilité d'apparition, dans le schéma suivant :



A chaque mise en attente d'une tâche correspondra ultérieurement un passage de "en attente" à "active". Ce passage se fera à la suite d'une fin d'entrée-sortie réelle et correspondra à une attente dont la probabilité d'arrivée était soit $\alpha_i \omega_i dt$ soit $\mu_i dt$. Par suite lorsqu'une entrée-sortie réelle se termine, elle entrainera avec une probabilité

$$\rho_i = \frac{\mu_i + \alpha_i \omega_i}{\omega}$$ le passage d'une tâche de l'état en attente à active et, avec une probabilité $1 - \rho_i$, elle ne changera l'état d'aucune tâche.

$$1 - \rho_i = \frac{(1-\alpha_i)\omega_i - \mu_i}{\omega_i}$$

on a bien $0 \leq 1 - \rho_i \leq 1$ car le nombre de mises en attente sans lancement d'entrée-sortie est nécessairement inférieur ou égal au nombre de lancements d'entrées-sorties réelles n'ayant pas entraîné de mise en attente d'une tâche :

$$\Rightarrow (1-\alpha_i)\,\omega_i \geq \mu_i$$

$$\Rightarrow 0 \leq (1-\alpha_i)\,\omega_i - \mu_i$$

$\omega_i > 0 \quad \Rightarrow 0 \leq \dfrac{(1-\alpha_i)\,\omega_i - \mu_i}{\omega_i}$

D'autre part :

$0 \leq 1 - \alpha_i \leq 1 \quad \Rightarrow (1-\alpha_i)\,\omega_i \leq \omega_i$

$\mu_i \geq 0 \quad \Rightarrow (1-\alpha_i)\,\omega_i - \mu_i \leq \omega_i$

$\omega_i > 0 \quad \Rightarrow \dfrac{(1-\alpha_i)\,\omega_i - \mu_i}{\omega_i} \leq 1$

par suite $1 - \rho_i$ et donc ρ_i sont bien compris entre 0 et 1.

Nous allons passer maintenant à l'évaluation pratique des μ_i et α_i pour laquelle le paramètre ρ_i nous sera utile.

La comptabilité ne peut guère nous être utile pour l'évaluation de ces paramètres qui n'ont guère d'intérêt pour le calcul du prix qu'un utilisateur paiera au Centre de Calcul à la fin du mois.

Aussi allons-nous une fois encore utiliser la méthode exposée en V.3.1.

Nous avons le choix, pour connaitre les α_i et μ_i, entre l'évaluation des α_i et μ_i ou celle des α_i et ρ_i, les ω_i étant supposés connus.

Pour les μ_i, il est nécessaire de piéger le régulateur alors que pour α_i et ρ_i, il faut piéger le superviseur d'entrées-sorties. Nous choisirons cette dernière solution.

Il faudra donc, lors de l'arrivée d'une demande d'entrée-sortie pour un organe de type i, aller repérer dans les tables du système si la tâche qui a lancé cette demande est ou non mise en attente. On disposera de deux compteurs par type d'organe et on incrémentera de 1 l'un si la réponse est positive et l'autre si elle est négative.

A la fin de la période d'évaluation, on déduira donc de manière immédiate les paramètres α_i comme les rapports du résultat contenu dans le premier compteur à la somme des résultats

des deux compteurs, pour chaque i.

Pour obtenir les ρ_i, on travaillera de manière analogue, mais en repérant les fins d'entrées-sorties et en cherchant dans les tables du système si l'entrée-sortie qui se termine est bien destinée à une tâche utilisateur, et non au système. Dans le premier cas, il faudra tester si cette entrée sortie va permettre à la tâche de passer de "en attente" à "active" ou non (la tâche peut en effet rester "en attente" ou rester "active").

V.3.2.3.5. σ, σ' et ω_b.

Ces trois paramètres ont pour caractéristique commune d'être représentatifs du fonctionnement du moniteur.

Les valeurs qu'ils prendront dépendront totalement du type de moniteur, de son importance et de sa structure. Les valeurs prises par σ et σ' sont en particulier essentielles dans la mesure où elles rendent compte de l'efficacité du système : ainsi un moniteur très sophistiqué mais qui consacre beaucoup de temps à lui-même peut être rejeté tout autant qu'un moniteur utilisant peu de temps de calcul mais rendant peu de services aux utilisateurs.

La répartition, entre parties résidente et non résidente, des différents modules du système va avoir un grand rôle sur le paramètre ω_o. Bien entendu cette répartition doit tenir compte de la charge de l'installation, pour être optimale.

Les utilisateurs joueront aussi un rôle important sur ω_o dans la mesure où le moniteur profite de la place qu'ils n'utilisent pas pour garder en mémoire certaines de ses parties non résidentes qui sont souvent appelées : il s'agit d'un swapping mais uniquement du moniteur et qui n'implique pas l'utilisation du swapping pour les travaux utilisateurs.

L'estimation de ces paramètres peut se faire en utilisant la comptabilité qui donne le temps pendant lequel

le moniteur a utilisé l'unité centrale pour ses besoins propres. D'autre part elle donne le nombre d'entrées-sorties disque rapide qu'il a lancées pendant cette période : on en déduit donc ω_0 immédiatement. Et si on suppose que σ et σ' sont égaux on en déduit aussi ce paramètre.

Néanmoins σ et σ' ne sont pas égaux a priori, et il semble très intéressant d'évaluer la valeur de chacun. En effet l'un (σ) représente du temps de calcul utilisé par le moniteur alors qu'il aurait pu être utilisé par des tâches-utilisateurs, c'est donc dans un certain sens une perte, en revanche l'autre (σ') représente du temps qui, étant donné l'organisation du système et sa charge, ne pouvait de toute façon être utilisé que par le moniteur puisqu'aucune tâche utilisateur n'était alors active.

Si la comptabilité ne permet pas d'évaluer séparément ces deux paramètres, il faudra les estimer en utilisant la méthode exposée en V.3.1.

Nous savons que chaque fois qu'une tâche devient active, ou passe en attente d'entrées-sortiesle moniteur remet à jour certaines de ses tables (ex : comptabilité). Il faudra donc disposer d'un piège dans le module de mise à jour des tables du moniteur. Ainsi, à chaque passage dans le module on pourra tester par un module de mesure ajouté au système s'il y a ou non des tâches actives en mémoire.

Supposons que l'on détecte lors d un passage dans le module de mesure qu'il n'y a plus de tâche active en mémoire, on range alors dans un compteur C1 l'heure en unités de temps, et dans un compteur C2 la valeur, trouvée dans les tables du système, du compteur du temps pendant lequel le moniteur a travaillé pour ses besoins propres depuis la dernière initialisation du moniteur.

Lors du prochain passage dans le module de mise à jour des tables du moniteur, s'il y a à nouveau une tâche active, on ira, grâce au module de mesure, chercher les nouvelles valeurs C'1 et C'2 de l'horloge et du

compteur d'activité du moniteur.On ajoutera C'1-C1 à un compteur B1 et C'2-C2 à un compteur B2. B1 et B2 ont été mis à zéro au début de la période d'évaluation.

A la fin de l évaluation on déduira:

$$\sigma' = \frac{B1}{B2}$$

On obtiendra σ de manière analogue.

Enfin, comme pour d'autres paramètres étudiés précédemment,il peut sembler, dans certains cas, intéressant d'évaluer ω_0, σ et σ' en fonction de n le nombre de travaux de la file V. Cela alourdira les mesures, mais peut facilement être envisagé.

Voyons le cas par exemple de l'évaluation de $\omega'(n)$. Le procédé est analogue à celui exposé précédemment pour le calcul de σ', mais on y ajoutera la consultation d'une table du système permettant de déterminer n le nombre de travaux actuellement dans V. . Et on incrémentera non les compteurs B1 et B2 mais des compteurs B1(n+1) et B2(n+1). A la fin de l'évaluation on aura :

$$\sigma'(n) = \frac{B1(n+1)}{B2(n+1)}$$

V.3.2.3.6. $\prod_{i=1}^{a} \pi_{n+1}(b_i)$

Nous avons a types disjoints de ressources, chacune en quantité b_i , $i = 1 \text{ --- } a$.

Ces ressources sont indépendantes les unes des autres.

Pour qu'un travail puisse entrer dans la file V, il fa qu'on puisse lui attribuer toutes les ressources qui lui sont nécessaires.

Par suite, cet évènement arrivera avec une probabilité $\prod_{i=1}^{a} \pi_{n+1}(b_i)$ si on note $\pi_{n+1}(b_i)$ la probabilité que n+1

travaux, les n présents dans la file V et le candidat au lancement , demandent au plus b_i éléments de la ressource i sachant que n travaux, les n présents dans la file V, en demandent au plus b_i.

Et on aura alors :

$$\pi_{n+1}(b_i) = \sum_{k_i=0}^{B_i} \pi'_n(k_i)\, \pi(b_i - k_i)$$

Dans cette formule $\pi(b_i - k_i)$ est la probabilité de demande par le travail candidat de $b_i - k_i$ unités de la ressource i au plus, cette probabilité ne fait pas, a priori, intervenir le temps d'allocation demandé.

En revanche $\pi'_n(k_i)$,qui est la probabilité pour que les n travaux de V disposent de l'allocation de k_i éléments de la ressource i, fait intervenir dans son évaluation la durée probable d'allocation. En effet, un travail qui demande 4 dérouleurs de bandes magnétiques pour une heure n'aura pas la même incidence sur le lancement des travaux qu'un travail qui en demande quatre pendant cinq minutes.

Bien entendu, il s'agit ici d'une formule générale qu'il faudra adapter aux différents types de régulateurs rencontrés.

Ainsi il existe des systèmes qui suppléent automatiquement à une surcharge de certains périphériques rapides en faisant passer, au moyen de changements de support , une partie de leur charge sur d'autres périphériques rapides moins sollicités.

On pourra par exemple envoyer sur disques lents des fichiers qui auraient dû normalement se trouver sur les disques rapides, quitte à les rappeler sur disques rapides lorsque ceux ci ne seront plus surchargés.

Il est certain que de tels aménagements doivent être considérés pour l'évaluation des $\pi_{n+1}(b_i)$, et aussi d'ailleurs pour celle des autres paramètres probabilistes du système.

De même si la file E est divisée en plusieurs sous-files de priorités différentes et telles que les $\pi(b_i)$ soient très différents de l'une à l'autre, il faudra peut-être, pour mieux coller à la réalité, envisager des $\pi^j(b_i)$ où j est le numéro de la sous file associée. Mais cela risque d'alourdir beaucoup le modèle.

Donnons un exemple où l'on aurait trois sous-files $E^{(1)}$, $E^{(2)}$ et $E^{(3)}$. La sous-file d'indice (j) ne peut-être consultée que si les sous-files d'indice inférieur à j sont vides.

Alors au lieu d'utiliser $(1-E_0(t)) \prod_i \pi_{n+1}(b_i)$

dans notre modèle nous devrions utiliser

$$(1-E_0^{(1)}(t)) \prod_i \pi_{n+1}^{(1)}(b_i) + E_0^{(1)}(t)(1-E_0^{(2)}(t)) \prod_i \pi_{n+1}^{(2)}(b_i)$$
$$+ E_0^{(1)}(t).E_0^{(2)}(t)(1-E_0^{(3)}(t)) \prod_i \pi_{n+1}^{(3)}(b_i)$$

Il pourra être alors délicat de déterminer $\pi'_n(b_i)$.

Ces probabilités dépendent donc du software, mais elles traduisent aussi des caractéristiques des travaux passant sur l'installation.

Pour les évaluer, la méthode la plus simple est une fois encore la comptabilité de l'exploitation, qui pour chaque travail répertorie les ressources demandées et le temps pendant lequel elles sont allouées.

La recherche des statistiques pour établir les paramètres $\pi_{n+1}(b_i)$ peut mettre en évidence une variation des paramètres au cours de la journée. Ce manque d'homogénéité peut venir d'une mauvaise adaptation du régulateur à sa charge avec, en fin de session, une dégradation des performances, quand vont enfin passer les travaux rejetés jusqu'alors pour coexistence difficile.

V.3.2.4. Conclusion.

Dans ce paragraphe V.3., nous avons indiqué comment on pouvait procéder pour évaluer les paramètres de notre modèle.

Des perfectionnements sont bien sûr possibles, et pour certains paramètres, comme nous l'avons indiqué pour les ω_i, on aurait intérêt, si le processus de mesure le permet, à différencier les valeurs obtenues pour les paramètres selon les types de travaux (ex : travaux scientifiques ou de gestion) ou de tâches (compilation, assemblages, éditions de liens etc...). On obtient alors la valeur du paramètre correspondant à l'ensemble en faisant la somme des résultats obtenus par types, pondérée par la fréquence relative des différents types. Ceci permet d'apprécier les variations des paramètres en fonction des variations de la charge en travaux ou des ressources software. Ce sera donc particulièrement intéressant si on envisage une optimisation ou si on cherche à évaluer l'impact de modifications de l'installation.

V.4. Conclusion

Pour conclure ce chapitre sur l'évaluation des paramètres nous pouvons dire :

- que la connaissance des paramètres nécessite la mise en oeuvre de différents moyens d'investigation simultanément.
- Ces moyens d'investigation sont variables selon la précision des paramètres que l'on veut obtenir. Certaines méthodes peuvent être très coûteuses en temps-machine et entraîner d'importantes perturbations au fonctionnement normal de l'installation.
- L'importante utilisation de la comptabilité met en évidence tout à la fois son intérêt et ses limites. Son intérêt puisqu'elle permet d'avoir, sans perturbation, des renseignements sur tous les travaux. Ses limites puisqu'elle est élaborée dans un but essentiellement "financier" de sorte que des renseignements très intéressants sur le système n'y figurent pas et d'autre part ses résultats sont très globaux ce qui limite beaucoup leur intérêt.

Finalement, étant donnés les problèmes posés par la mise en place de mesures sur un système par les utilisateurs, on voit tout l'intérêt de modules de mesure intégrés au moniteur dès l'origine, permettant à l'utilisateur une évaluation aisée et précise des paramètres qui l'intéressent afin qu'il puisse modifier certaines caractéristiques de son installation en fonction de sa charge et au fur et à mesure de l'évolution de celle ci.

CONCLUSION

CONCLUSION :

Le modèle stochastique que nous avons établi ne prend tout son intérêt que lorsque les paramètres qui lui permettent d'être appliqué sont évalués.

Des travaux dans ce sens ont été faits à Nancy sur le C.I. 10.070 de l'Institut Universitaire de Calcul Automatique : d'une part on a essayé d'implanter une méthode de mesure par échantillonnage (cf M.F. Pistré, Réf. [69]), et d'autre part une méthode basée sur la trace a été élaborée pour étudier les demandes de pages qui se produiraient si l'on utilisait le changement en mémoire de pages à la demande.

Il est apparu très clairement tout au long de cette étude que le problème de l'évaluation de systèmes d'exploitation (et les problèmes qui y sont liés : modification, comparaison, optimisation) est un problème qui doit être résolu pour chaque installation, en fonction de ses caractéristiques propres et de sa charge en travaux.

Cette constatation montre de quel intérêt peut être notre modèle dont la souplesse permet une adaptation à chaque installation.

APPENDICE

APPENDICE

LES PHENOMENES D'ATTENTE

Nous allons étudier dans cet appendice certains aspects probabilistes de la théorie des phénomènes d'attente qui nous sont utiles pour l'élaboration de notre modèle de système d'exploitation.

I. PROCESSUS DE POISSON :

Nous allons démontrer que, sous certaines conditions, la loi d'arrivée d'unités dans une file d'attente suit un processus de Poisson.

Soit N le nombre d'arrivées dans l'intervalle de temps $[0, t]$, et $P_n(t)$ la loi de N. Nous faisons les hypothèses suivantes :

i) $P_n(t)$ ne dépend que de l'intervalle de temps et ne dépend pas de l'instant initial (homogénéité ou stationnarité dans le temps).

ii) La probabilité que l'arrivée d'une unité se produise plus d'une fois dans l'intervalle de temps dt est infiniment petite par rapport à dt.

iii) La probabilité que l'arrivée d'une unité se produise dans l'intervalle de temps dt, dt étant très petit, est λdt.

Nous allons montrer qu'alors

$$P_n(t) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

Démonstration :

Nous savons que $\sum_{i \geq 0} P_i(dt) = 1$ car les éventualités s'excluent mutuellement. Comme $P_1(dt) = \lambda dt$ et $\sum_{i \geq 2} P_i(dt)$ est très

petit par rapport à dt, par hypothèse , alors $P_0(dt) = 1 - \lambda dt$.

L'hypothèse i) nous permet alors d'établir une relation de récurrence :

$$\cdot\ P_0(t+dt) = P_0(t).P_0(dt) = P_0(t)\,(1-\lambda dt)$$

$$\Rightarrow P_0(t+dt) - P_0(t) = -\lambda P_0(t)dt$$

$$\Rightarrow \frac{d}{dt}P_0(t) = -\lambda P_0(t) \qquad (1)$$

. et pour $n > 0$

$$P_n(t+dt) = P_n(t).P_0(dt) + P_{n-1}(t).P_1(dt)$$

$$\Rightarrow \frac{d}{dt}P_n(t) = \lambda P_{n-1}(t) - \lambda P_n(t) \qquad (2)$$

Introduisons alors la fonction caractéristique des $P_n(t)$: $\varphi(\tau, t) = \sum_{n=0}^{\infty} P_n(t)\, e^{i\tau n}$. Les relations (1) et (2) nous donnent alors :

$$\frac{d}{dt}\varphi(\tau, t) = (\lambda e^{i\tau} - \lambda)\,\varphi(\tau, t)$$

$$\Rightarrow \varphi(\tau, t) = C(\exp\left[\lambda t(e^{i\tau}-1)\right]$$

mais $\varphi(0, t) = 1$, donc $C = 1$ et on reconnait la fonction caractéristique d'une loi de Poisson de paramètre λt, par suite :

$$P_n(t) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

II. LOI DES INTERVALLES DE TEMPS SEPARANT DEUX ARRIVEES CONSECUTIVES.

Nous allons étudier la variable aléatoire $\Theta = t_{i+1} - t_i$ où t_i est l'instant d'une arrivée.

$$P_r\left[\theta < \Theta \leq \theta + d\theta\right] = P_0(\theta).P_1(d\theta)$$

Comme $P_n(t) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$ $\qquad n \geq 0$

$P_0(\theta) = e^{-\lambda\theta}$ et $P_1(d\theta) = \lambda d\theta$

par suite $P_r\left[\theta < \Theta \leq \theta + d\theta\right] = e^{-\lambda\theta}\lambda\, d\theta$ et

la densité de Θ est donc $\lambda e^{-\lambda\theta}$

Θ suit donc une loi gamma de paramètre 1 ou loi exponentielle.

III. ETUDE D'UNE FILE D'ATTENTE A UNE STATION :

1) Mise en équation :

Nous allons chercher la distribution de la variable aléatoire N représentant le nombre d'unités dans la file, c'est-à-dire en attente et en cours de service.

Nous nous placerons dans le cas très usuel ou les arrivées suivent la loi de Poisson et les intervalles de temps de service la loi exponentielle. Les hypothèses seront donc :

- La probabilité qu'une unité arrive dans la file dans l'intervalle de temps dt est λdt ; λ est le nombre moyen d'arrivées par unité de temps.
- La probabilité que la fin de service d'une unité se produise dans l'intervalle dt est μdt ; $1/\mu$ est l'intervalle de temps moyen de service.
- La probabilité de plusieurs arrivées ou de plusieurs fins de service dans l'intervalle dt est infiniment petite par rapport à dt.

Cherchons la probabilité $P_n(t+dt)$ d'avoir n unités, $n > 0$, dans la file à l'instant $t + dt$. Elle s'exprime comme la somme des quatre probabilités indépendantes composées suivantes :

1) Le produit des probabilités qu'il y ait :

a) n unités dans la file à l'instant t	$P_n(t)$
b) aucune arrivée entre t et $t + dt$	$1-\lambda dt$
c) aucune fin de service dans l'intervalle dt	$1-\mu dt$

2) Le produit des probabilités qu'il y ait :

a) n+1 unités à la file à l'instant t	$P_{n+1}(t)$
b) une fin de service entre t et t+dt	μdt
c) aucune arrivée entre t et t+dt	$1 - \lambda dt$

3) Le produit des probabilités qu'il y ait :

a) n-1 unités dans la file à l'instant t	$P_{n-1}(t)$
b) aucune arrivée entre t et t+dt	λdt
c) aucune fin de service entre t et t+dt	$1-\mu dt$

4) Le produit des probabilités qu'il y ait :

a) n unités dans la file à l'instant t	$P_n(t)$
b) une arrivée entre t et t+dt	λdt
c) une fin de service entre t et t+dt	μdt

Par suite

$$P_n(t+dt) = P_n(t)\,(1-\lambda dt)\,(1-\mu dt) + P_{n+1}(t)\,(\mu dt)\,(1-\lambda dt) + P_{n-1}(t)\,(\lambda dt)\,(1-\mu dt) + P_n(t)\,(\lambda dt)\,(\mu dt)$$

Si nous éliminons les infiniment petits d'ordre supérieur à 1 il vient alors :

$$P_n(t+dt) = P_n(t)\,(1-\lambda dt)-\mu dt) + P_{n-1}(t)\,\lambda dt + P_{n+1}(t)\,\mu dt$$

$$\Rightarrow \frac{P_n(t+dt) - P_n(t)}{dt} = -P_n(t)\,(\lambda+\mu) + P_{n-1}(t)\,\lambda + \mu P_{n+1}(t)$$

$$\Rightarrow \boxed{\frac{d}{dt} P_n(t) = \lambda P_{n-1}(t) + \mu P_{n+1}(t) - (\lambda+\mu)P_n(t)} \qquad (1)$$

pour $n > 0$

Dans le cas où n = 0 la probabilité qu'il y ait 0 unité dans la file au temps t + dt est la somme des deux probabilités indépendantes composées suivantes :

1) Le produit des probabilités qu'il y ait :

a) aucune unité dans le système au temps t	$P_0(t)$
b) aucune arrivée dans l'intervalle dt	$1-\lambda dt$

2) Le produit des probabilités qu'il y ait :

a) une unité dans le système au temps t	$P_1(t)$
b) aucune arrivée dans l'intervalle dt	$1-\lambda dt$
c) une fin de service dans l'intervalle dt	μdt

Donc :

$$P_0(t+dt) = P_0(t)\,(1-\lambda dt) + P_1(t)\,(1-\lambda dt)\,(\mu dt)$$

$$\Rightarrow \boxed{\frac{d}{dt} P_0(t) = -\lambda P_0(t) + \mu P_1(t)} \qquad (2)$$

b) Résolution en régime permanent :

Nous allons chercher les fonctions $P_n(t)$ qui constituent la solution du système différentiel (1), (2) dans le cas particulier où les probabilités P_n sont indépendantes de t.

On a alors $P'_n(t) = 0$ et $P_n(t) = P_n$, le processus est dit stationnaire et permanent. Le système d'équations se réduit à :

$$\begin{cases} \lambda P_{n-1} + \mu P_{n+1} - (\lambda+\mu)P_n = 0 & n > 0 \\ -\lambda P_0 + \mu P_1 = 0 \end{cases}$$

De plus par définition, nous avons $\sum_{i=0}^{\infty} P_i = 1$

En procédant par récurrence on obtient :

$$P_1 = \frac{\lambda}{\mu} P_o \qquad \text{et} \qquad P_n = \left(\frac{\lambda}{\mu}\right)^n P_o \quad \text{pour } n > 0$$

Comme $\sum_{n=0}^{\infty} P_n = 1$

$$\Rightarrow P_o \sum_{n=o}^{\infty} \left(\frac{\lambda}{\mu}\right)^n = 1$$

L'hypothèse $\lambda / \mu < 1$ est donc nécessaire pour assurer la convergence de la série et on a alors :

$$\sum_{i=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^n = \frac{1}{1-\frac{\lambda}{\mu}} \Rightarrow P_0 = 1 - \frac{\lambda}{\mu}$$

$$\Rightarrow \boxed{P_n = \left(\frac{\lambda}{\mu}\right)^n \left(1 - \frac{\lambda}{\mu}\right)} \quad n \geq 0$$

c) Résolution en régime transitoire :

Posons $\psi = \lambda / \mu$ (c'est l'intensité du trafic) et prenons comme conditions initiales $P_0(0) = 1$ et $P_n(0) = 0$ pour $n > 0$. La solution du système (1), (2) est alors :

$$P_n(t) = e^{-(\lambda+\mu)t}\left[\psi^{n/2} I_{-n}(2\sqrt{\lambda\mu}\,t) + \psi^{(n-1)/2} I_{n+1}(2\sqrt{\lambda\mu}\,t)\right.$$

$$\left. + (1-\psi)\,\psi^n \sum_{k=n+2}^{\infty} \psi^{-(k/2)} I_k(2\sqrt{\lambda\mu}\,t)\right]$$

où $I_n(x) = j^{-n} J_n(jx)$

est la fonction de Bessel modifiée de seconde espèce d'ordre n.

La complexité de cette formule explique que nous n'ayons pas pu étudier le régime transitoire dans notre modèle qui, rappelons-le, fait intervenir plusieurs files d'attente interdépendantes.

BIBLIOGRAPHIE

BIBLIOGRAPHIE - ARTICLES

1. IGAL ADIRI.
   a)"A dynamic time-sharing priority queue"
   J.A.C.M, Vol 18, n°4, October 1971, pp 603-610.
   b)"A note on some mathematical models of time-sharing systems".
   J.A.C.M, Vol 18, n°4, October 1971, pp 611-615.

2. R.ASLANIAN et M.BENNETT.
   "Modelage évolutif et évaluation d'ordinateurs et de systèmes d'exploitation"
   R.A.I.R.O, 6°année, B1, 1972, pp 39-52.

3. BERNSTEIN A.J et SHARP J.C.
   "A policy-driven scheduler for a time-sharing system".
   C.A.C.M, Vol 14,n°2,Feb.1971, pp 74-78.

4. J. BLATNY, S.R. CLARK and T.A. ROURKE.
   "On the optimization of performance of time-sharing systems by simulation".
   C.A.C.M. Vol 15, June 1972, n°6, pp 411-420.

5. WILLIAM H.BURGE and ALAN G. KONHEIM.
   "An accessing model".
   J.A.C.M. Vol 18, n°3, July 1971, pp 400-404.

6. G.J. BURNETT and E.G. COFFMAN, Jr.
   "A combinatorial problem related to interleaved memory systems".
   J.A.C.M. Vol 20, n°1, January 1973, pp 39-45.

7. CALINGAERT PETER.
   "System performance evaluation : survey and appraisal".
   C.A.C.M. Vol 10, n°1, January 1971, pp 12-18.

8. CHENG P.S.
   "Trace driven system modeling".
   IBM SYSTEM JOURNAL n°4, 1969, pp 280-289.

9. COFFMAN E.G, Jr.
   "Analysis of two time-sharing algorithms designed for limited swapping".
   J.A.C.M. Vol 15, n°3, July 1968, pp 341-353.

10. COFFMAN E.G, Jr and THOMAS A. RYAN Jr.
"A study of storage partitionning using a mathematical model for locality".
C.A.C.M. Vol 15, n°3, March 1972 pp 185-190.

11. PETER J.DENNING and STUART C. SCHWARTZ.
"Properties of the working-set model".
C.A.C.M. Vol 15, n°3, March 1972 pp 191-198.

12. G. DREAN.
"La multiprogrammation : mythes et réalités".
R.I.R.O. 2° année, n°9, 1968, pp 97-116.

13. M.E. DRUMMOND, JR.
"A perspective on system performance evaluation".
IBM SYST.J, n°4, 1969, pp 252-263.

14. EVERETT J.L.
"State probabilities in congestion problems characterized by constant holding time".
J. Opus.Res.Soc.Amer, 1, n°5, pp 279-285, Nov.1953.

15. JAMES D.FOLEY
"A markovian model of the University of Michigan Executive system".
C.A.C.M. Vol 10, n°9, September 1967, pp 584-588.

16. GAVER D.P.
"The influence of servicing times in queuing processes".
J. Opus.Res.Soc.Amer.,2,n°2, pp 139-149, May 1954.

17. GAVER D.P, Jr.
"Probability models for multiprogramming computer systems".
J.A.C.M. Vol 14, n°3, July 1967, pp 423-438.

18. DONALD P. GAVER.
"Analysis of Remote terminal backlogs under heavy demand conditions".
J.A.C.M. Vol 18, n°3, July 1971, pp 405-415.

19. HARRY C. HEACOX, Jr and PAUL W.PURDOM, Jr.
"Analysis of two time-sharing queuing models".
J.A.C.M. Vol 19, n°1, January 1972, pp 70-91.

20. HELLERMAN H.
"Some principles of time-sharing scheduler strategies".
IBM SYST.J. n°2, 1969, pp 94-117.

21. K. HUTCHINSON.
"Computer center simulation project"
C.A.C.M. Vol 8, n°9, September 1965, pp 559-568.

22. JACKSON R.R.P.
"Queuing systems with phase type service".
Operating Res.Quart,5, n°4, pp 109-120, Déc.1954.

23. STEPHEN R. KIMBLETON.
"The note of computer system models in performance evaluation".
C.A.C.M. July 1972. Vol 15. n°7. pp 586-590.

24. LEONARD KLEINROCK.
"Swap-time considerations in time-shared systems".
IEEE Transactions on computers Vol C.19 n°6. June 1970 pp 534-540.

25. DAVID G. KENDALL
"On the code of variable generation time in the development of a stochastic birth process".
Biometrika 35. pp 316-330, Dec.1948.

26. DAVID G. KENDALL.
"Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded Markov chain".

27. ALAN G. KONHEIM and BERND MEISTER
"Service in a loop system".
J.A.C.M. Vol 19, n°1, January 1972, pp 92-108.

28. C.L.LIU and JAMES W.LAYLAND.
"Scheduling algorithms for multiprogramming in hard-real-time environment".
J.A.C.M. Vol 20, n°1, January 1973, pp 46-61.

29. W.C. LYNCH
"Operating system performance".
C.A.C.M. July 1972, Vol 15, n°7, pp 579-585.

30. M.H. MACDOUGALL.
"Computer system simulation : an introduction"
Computing Surveys Vol 2, n°3, Sept 70, pp 19?-209.

31. J.M. MACKINNEY.
"Survey of analytical time-sharing models"
Computing surveys Vol 1, n°2, June 1969, pp 105-116.

32. F.R. MOORE.
"Computational model of a closed queuing network with exponential servers".
IBM J.Res.Develop. Vol 16, n°6, November 1972, pp 567-572.

33. MANOU - ENICA Toulouse.
"Evaluation des performances des systèmes"
Exposé à l'école d'été d'Informatique - Alès - Eté 1971.

34. PHILIP M. MORSE.
"Stochastic properties of waiting lines".
J.Opus.Res.Soc.Amer.,3, n°3, pp 255-261, Aug.1955.

35. ALAN G. NEMETH and PAUL D. ROVNER.
"User programm measurement in a time-shared environment".
C.A.C.M. Oct.1971, Vol 14, n°10, pp 661-666.

36. NORMAN R.NIELSEN.
"The simulation of time-sharing systems".
C.A.C.M. Vol 10, n°7, July 1967, pp 397-412.

37. NORMAN R.NIELSEN.
"An analysis of some time-sharing techniques".
C.A.C.M. Feb.1971.Vol 14, n°2, pp 79-90.

38. P.H. ODEN and G.S. SHELDER.
"A model of memory contention in a paging machine".
C.A.C.M. Aug. 1972. Vol 15. n°8, pp 761-771.

39. TAD. B. PINKERTON.
"Performance monitoring in a time-sharing system".
C.A.C.M. Vol 12, n°11, Nov 1969, pp 608-610.

40. B. POUSSOT I.M.A.G , Grenoble.
"Mesures de performances de systèmes d'exploitation".
Exposé du 13.11.1970.

41. M.T.RAYNAUD.
"Les mesures dans un système d'exploitation partagé".
R.I.R.O., B1, 1969, pp 119-126.

42. G. REINIER.
"Contribution à l'étude statistique du fonctionnement du calculateur CDC 6600".
R.I.R.O ,B2 ,1970, pp 17-34.

43. X. ROUSSET de PINA.
"Modèle déterministe pour l'étude de l'efficacité d'un système en temps partagé".
R.I.R.O,B3,1971, pp 109-121.

44. J.L. SMITH.
"An analysis of time-sharing computer systems using Markov models".
Proceedings Spring Joint Computer Conference 1960,pp 87-95.

45. JOHN L. SMITH.
"Multiprogramming under a page on demand strategy".
C.A.C.M. Vol 10, n°10, Oct.1967, pp 636-646.
et Vol 13, n°4 , Apr.1970, pp 265.

46. THOMAS R. SPACEK.
"A proposal to establish a pseudo-virtual memory via writable overlays".
C.A.C.M. June 1972, Vol 15, n°6, pp 421-426.

47. SAUL STIMLER.
"Some criteria for time-sharing system performance".
C.A.C.M. Vol 12, n°1, 1969, pp 47-53.

48. TOBY J. TEOREY and TAD.B. PINKERTON.
"A comparative analysis of disk scheduling policies".
C.A.C.M. March 1972,Vol 15, n°3, pp 177-184.

49. WAH CHUN CHAN.
"A computer processing queuing system with feedback".
Information and Control, 16, 1970, pp 473-486.

50. V.L. WALLACE and D.L. MASON.
"Degree of multiprogramming in page on demand systems".
C.A.C.M. VOL 12, n°6, June 1969, pp 305-318.

51. Compte-rendu d'activité du groupe Evaluation de systèmes pour l'année 1967.
R.I.R.O. 2° année, n°12, 1968, pp 89-103.

52. IRIA - 19.3.1971.
Journées "Mesures et simulation de système d'exploitation".
Mme J. RAYMOND : "CASSCOU, un modèle de simulation de systèmes conversationnels".
Melle E. MIDROUILLET : "Langages de simulation et simulation de systèmes".
M.ASLANIAN et BENNETT :"Modelage évolutif et évaluation de systèmes d'exploitation".
M. de SAINT-SEINE :"Méthodes de mesures de systèmes".
M. FRAVAL : " "Mesures de performances de systèmes".

53. AFCET 13.10.1971.
Journée "Outils de mise au point et d'optimisation des systèmes d'exploitation".
M de FERRAN :"Tour d'horizon des techniques de mise au point et d'optimisation des systèmes d'exploitation"
M. BARBOTTIN :"Module de gestion de fichiers pour l'intégration de systèmes".
M.DESACHY et DELMOTTE :"Possibilités offertes par un système générateur de machines virtuelles dans les domaines de la mise au point de la mesure des systèmes d'exploitation".
Mme THURY : "AMAP": cutil d'optimisation de l'OS-360.
M.BUISSON:"Evolution d'un outil de mise au point pour la réalisation de SIRIS 7/8".
M.DUCASSE:"Outils utilisés pour la mise au point du software de base sur la série P800 Philips".
M.GUEZ:"Comment rationaliser une exploitation sous DOS : Megam".

54. Communications of the A.C.M.
Vol 11, n°5, May 1968, pp 297-377.
"Operating system principles" : 10 articles.
BRANDELL and C.J. KUEHNER : "Dynamic storage allocations systems".
ROBERT C. DALEY and JACK B. DENNIS : "Virtual memory, processes and sharing in MULTICS".
G. OPPENHEIMER and N. WEIZER : "Ressource management for a medium scale time-sharing operating systems".
PETER J. DENNING : "The working set model for program behavior".
KURT FUCHEL and SIDNEY HELLER : "Considerations in the design of a multiple computer system with extended core storage".
EDSGER W.DIJKSTRA : "The structure of the "THE" multiprogramming system".
BUTLER W. LAMPSON : "A scheduling philosophy for multiprocessing systems".
EARL C. VAN HORN : "Three criteria for designing computing systems to facilitate debugging".
ROBERT M.G.GRAHAM : "Protection in an information processing utility".
JACK B. DENNIS : "A position paper on computing and communications".

55. E.N.I.C.A. Toulouse.
"Software d'estimation de programmes"
Projet LTP. 1971.

56. Documentation C.I.I.
Concernant C.I.I. 10070 et IRIS 80.

57. Documentation IBM
Sur séries 360 et 370.

LIVRES et THESES

58. Richard W. WATSON.
"Time sharing system design concepts".
1970. Mac GRAWHILL INC. USA.

59. M.V. WILKES, F.R.S.
"Time sharing computer systems".
3° Ed. 1970. Mac Donald LONDON and American Elsevier INC.
New York.

60. Charles JORDAN.
"Calculus of finite differences".
2° Ed 1950. Chelsea publishing Company - New York.

61. **Joelle** COUTAZ. Thèse de 3° cycle.
"Simulation d'un système conversationnel".
4.12. 1970. Grenoble.

62. Josette MAUBLANC. Thèse de 3° cycle.
"Etude statistique d'un système en temps partagé".
25.2.1970. Clermont-Ferrand.

63. William FELLER.
"An introduction to probability theory and its applications".
Volumes 1 et 2 - Wiley International Edition London.

64. Jean LEGRAS.
"Méthodes et techniques de l'analyse numérique".
1971. Dunod - PARIS.

65. C.W. CHURCHMAN, R.L. ACKOFF, E.L. ARNOFF.
"Eléments de recherche opérationnelle".
Dunod. PARIS. pp 355-434.

66. A. KAUFMANN.
"Méthodes et modèles de la recherche opérationnelle".
Dunod. PARIS. 1959.

67. A.O. GUELFOND.
"Calcul des différences finies".
Dunod. PARIS.

68. KOSAKU YOSIDA.
"Equations différentielles et intégrales".
Dunod. PARIS.

69. M.F. PISTRE.
"Rapport de D.E.A. Juin 1972. Université de Nancy I."
"Projet de réalisation d'un système de programmétrie sur
C.I.I. 10070 sous Bath-Processing Monitor".

70. A. KAUFMANN et R. CRUON.
"Les phénomènes d'attente. Théorie et applications"
Dunod. PARIS. 1961.

Abréviations :

C.A.C.M. Communications of the Association for Computing Machinery.

J.A.C.M. Journal of the Association for Computing Machinery.

R.I.R.O. Revue française d'informatique et de Recherche Opérationnelle.

NOM DE L'ETUDIANT : JOMIER Geneviève

NATURE DE LA THESE : Doctorat de Spécialité en Mathématiques Appliquées

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 22 Juin 1973

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

J.R. HELLUY