Revue Française de Gestion Industrielle
Vol. 16, n° 1

SYSTEMES DE PRODUCTION & SYSTEMES INFORMATIQUES

Romain Eliot*

Résumé. - Les systèmes informatiques actuels sont de natures très diverses et il est difficile d'en dresser une typologie. Néanmoins, deux critères peuvent être utilisés à des fins de classification: le nombre de machines utilisées et le besoin en temps de calcul. Nous nous intéresserons dans cette étude aux applications effectuant des *traitements complexes* et permettant le *travail simultané* de plusieurs utilisateurs. Par *traitement complexe*, on entend qu'un traitement demandé par un utilisateur prend un temps non négligeable, découpable en succession de traitements partiels, ponctués d'éventuelles interventions de l'utilisateur. Les différentes théories des systèmes de production sont applicables à de tels systèmes, et leur application fait l'objet de cette étude. L'engorgement d'un gros système informatique est souvent résolu en augmentant la puissance de calcul et en optimisant les différents composants du système, indépendamment les uns des autres, ce qui est souvent très coûteux. L'objet de cette étude est de proposer une approche globale pour l'optimisation des systèmes informatiques.

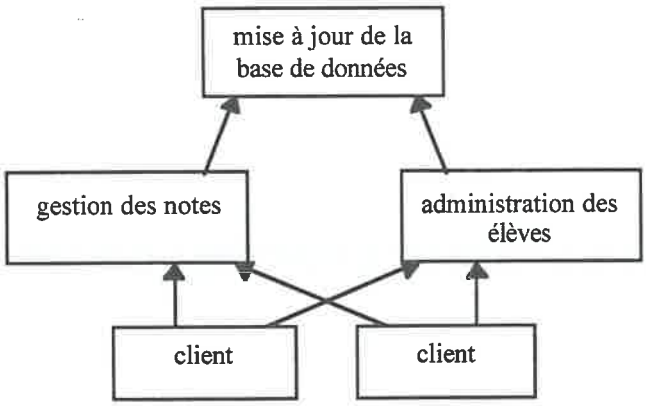
Mots clés : processus, gestion de bases, SPT, ordonnancement.

1. Description du type de système étudié

1.1 Approche globale

Cette étude traite des systèmes informatiques effectuant du traitement de données par processus. Ces systèmes présentent des problèmes semblables à ceux d'une usine en job shop. En effet, un ensemble de processus (programmes), chacun spécialisé dans une tâche, collabore en s'échangeant des données et des messages.

* Ingénieur ISIA-EMP



système informatique d'administration d'une école
Ce système est composé de deux processus clients, utilisés par le personnel de l'école pour émettre des requêtes, qui vont être adressées soit au processus de gestion des notes, soit au processus d'administration. Chacun de ces deux processus émet ensuite une requête de mise à jour de la base, envoyée au processus de gestion de la base.

Fig. 1 : exemple de traitement par processus

On peut distinguer plusieurs catégories de processus:

- certains effectuent des traitements: ils reçoivent une requête (description de la tâche à réaliser), effectuent le traitement et attendent l'arrivée de la requête suivante.
- d'autres sont chargés de l'organisation du travail. La plupart des systèmes peuvent être gérés sous forme d'un workflow, c'est à dire un graphe orienté, d'où la similitude avec une usine organisée en job shop.
- enfin certains processus sont chargés de la gestion des données, généralement stockées dans un système de base de données. Ces processus ont été distingués des processus de traitements car d'une part ils n'ont pas la même fonction, et de plus nous verrons qu'ils ne posent pas les mêmes problèmes. Néanmoins ces deux catégories pourront être confondues sous certaines hypothèses, pour simplifier.

Enfin il convient de noter que ces différents processus peuvent communiquer via l'intermédiaire d'un réseau, de fichiers ou d'une base de données.

1.2 Description détaillée

Il s'agit ici de décrire chacune des catégories de processus, en donnant notamment les renseignements suivants:

- tâche réalisée
- ordre de grandeur des paramètres décrivant le système
- types de problèmes rencontrés

1.2.1 Processus gérant le stockage des données

Ces processus sont en général liés à une base de données. La base de données est utilisée dans un gros système pour, bien sûr, stocker les données traitées et gérées par le système, mais aussi les données internes au système telles que la liste des tâches en cours, l'état d'avancement de chacune des tâches...

La lecture - et encore plus l'écriture - de données sur disque magnétique sont des opérations longues, durant au minimum 10 millisecondes, auxquelles s'ajoute un temps proportionnel à la taille des données lues.

Il est important de garder à l'esprit que la lecture et l'écriture de données sur support magnétique sont les activités qui ralentissent le système.

1.2.2 Processus effectuant des traitements

Les processus effectuant des traitements sont de natures très diverses. Ils reçoivent des requêtes, vont lire les données nécessaires (auprès d'un processus de lecture en base de données), effectuent leurs traitements puis vont écrire les résultats dans un stockage permanent, de nouveau auprès d'un processus gestionnaire de base de données. Les temps de traitement sont bien sûr extrêmement variables. Il convient de tenir compte des temps de lecture/écriture des données, car ceux-ci peuvent être non négligeables (notamment dans le cas de lecture d'images volumineuses, ou de lecture à travers un réseau lent). Durant les opérations de lecture / écriture, gérées par un processus de base de données, le processus de traitement est inactif.

1.2.3 Processus gérant le séquencement des tâches

On distinguera deux types de processus à l'intérieur de cette catégorie. En premier, ceux qui émettent les requêtes, en fonction des demandes des utilisateurs et des résultats des traitements précédents. Ensuite, il faut considérer les processus stockant les requêtes qui ne sont pas en cours de traitement et qui les fournissent en temps voulu aux serveurs de traitement. Ce sont ces derniers processus qui serviront à mettre en oeuvre les politiques d'optimisation de la production que nous allons étudier ci-après. Les temps de traitements de ces processus sont souvent quasi nuls, et nous n'en tiendrons pas compte par la suite. Certains systèmes n'ont pas de tels processus, car les processus de traitements déterminent alors eux-mêmes les requêtes à émettre à la suite de leur traitement ainsi que leurs destinataires respectifs.

1.2.4 Communications inter-processus

Les communications entre processus sont de manière générale assurées par un réseau. La rapidité de communication via un réseau dépend fortement de la charge du réseau, c'est pourquoi nous supposerons par la suite que le réseau ne ralentit pas les traitements, comparativement à un accès à la base de données. L'envoi d'un message d'une machine à une autre dure à peu près 1 milliseconde.

1.3 Hypothèses pour la suite de l'étude

- On considérera que c'est toujours la base de données qui peut ralentir le système, (en constituant un goulot d'étranglement), et non le réseau ou des processus de traitements. En effet, supposer que le réseau ralentit le système conduirait au même type d'étude, avec des temps de communication aléatoires, donc difficilement quantifiables.
- D'autre part, un processus de traitement constituant un goulot potentiel, peut être réparti sur plusieurs machines et ne plus représenter un risque de goulot, grâce à l'augmentation de la puissance de calcul qu'on lui fournit.

- On considérera que les temps de calculs des processus de traitement sont constants, alors qu'en pratique ils sont légèrement fluctuants.

2. Mise en évidence des limites d'un ordonnancement

2.1 *Etude de quelques exemples*

2.1.1 Un processus de gestion de base, un processus de traitement

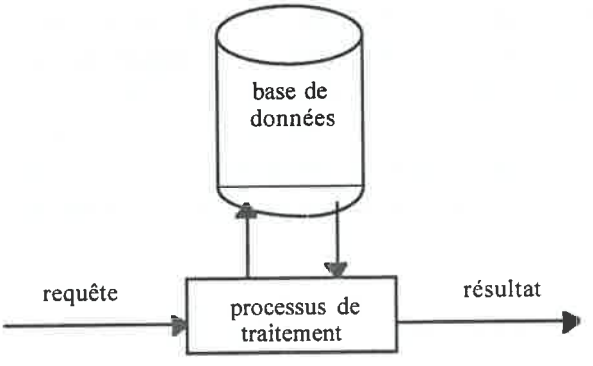


Fig. 2 : Exemple d'un processus de traitement

Tout d'abord, considérons le cas d'un système à deux processus: un processus de gestion de la base, et un processus de traitement. On cherche à minimiser le temps moyen passé dans le système pour chaque requête. On fait l'hypothèse que le temps de traitement de chaque requête est proportionnel au temps de lecture/écriture dans la base de données, ce qui est une hypothèse souvent vérifiée en pratique, pour un traitement donné. Si T_i est le temps total de traitement de la requête i (traitement propre de la requête + lecture / écriture en base), et en remarquant qu'à l'optimum le processus de traitement sera occupé en permanence, on conclut que le problème est équivalent à ordonnancer n requêtes sur un système à 1 machine.

De manière évidente, on minimise le temps moyen de séjour des requêtes par la règle SPT (Shortest Processing Time first). Ce cas est contraire à l'hypothèse de départ, mais il est utile de le garder à l'esprit car les cas suivants seront dérivés de celui-ci. En effet nous conserverons un processus de gestion de base, mais avec plusieurs processus de traitement.

2.1.2 Un processus de gestion de base, beaucoup de processus de traitement

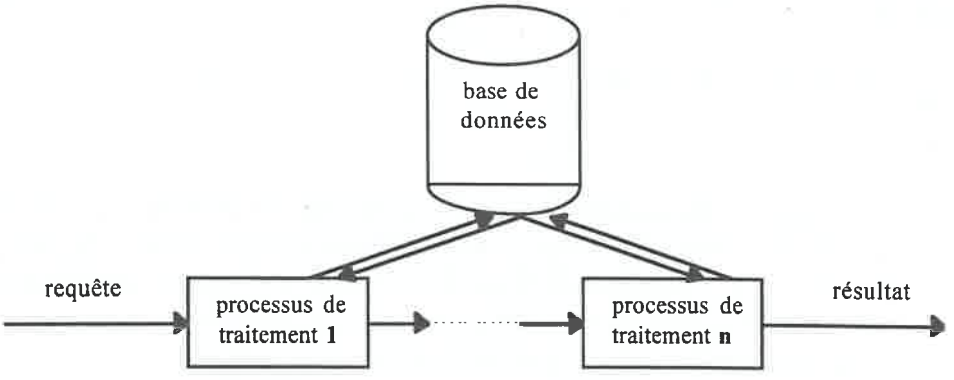


Fig. 3 : exemple de procesus de traitement

On entend par "beaucoup" un nombre suffisamment grand pour supposer qu'il n'y a jamais de requête en attente d'un processus de traitement. Par contre les processus de traitement seront certainement en attente de la base de données.

En conservant l'hypothèse que les traitements ont une durée proportionnelle à la durée de lecture / écriture dans la base, avec le même coefficient de proportionnalité pour tous les processus de traitement, nous pouvons tenter de minimiser le temps moyen de séjour d'une requête dans le système.

Hypothèses:

- il y a n requêtes à ordonnancer
- chaque requête doit subir un traitement de durée T_i, où i est le numéro de la requête, dont un temps $\tau.T_i$ de lecture en base au début de l'exécution.

Soit $\overline{F}_A$ le temps moyen de séjour à l' instant t_A, après le passage de n requêtes.
Soit T_k le temps de traitement de la requête k, et τT_k le temps
d' occupation de la base par le traitement de la requête k ($0 < \tau < 1$).
Soient i et j les deux requêtes suivant la requête k;
le temps moyen de passage après les requêtes i et j est :

$$(1)\ \overline{F}_B = \frac{1}{n+2}\left(n \times \overline{F}_A + (T_i) + (\tau T_i + T_j)\right)$$

En permutant l' ordre de i et j,

$$(2)\ \overline{F'}_B = \frac{1}{n+2}\left(n \times \overline{F}_A + (T_j) + (\tau T_j + T_i)\right)$$

$$\overline{F'}_B > \overline{F}_B \Leftrightarrow T_i < T_j$$

On constate donc que la minimisation du temps moyen de séjour dans le système est obtenue par l'application de la règle SPT.

De manière plus générale, on peut s'affranchir de l'hypothèse sur la proportionnalité du temps de traitement et du temps de lecture en base.

Soient ti les temps de lecture /écriture en base de la requête i.

Les équations (1) et (2) deviennent alors :

(1) $\overline{F}_B = \frac{1}{n+2}\left(n \times \overline{F}_A + (T_i) + (t_i + T_j)\right)$

(2) $\overline{F}_B = \frac{1}{n+2}\left(n \times \overline{F}_A + (T_j) + (t_j + T_i)\right)$

$\overline{F}_B > \overline{F}_B \Leftrightarrow t_i < t_j$

On constate que l'ordonnancement optimal est encore obtenu par la règle SPT, appliquée aux temps de lecture / écriture en base. L'optimisation globale du système passe donc par l'optimisation de son goulot, la base de données. On observe de manière évidente que le temps gagné sur les accès en base est gagné pour l'ensemble du système. L'approche OPT semble donc avoir un intérêt non négligeable pour notre système.

2.2 *Conclusions sur l'ordonnancement*

Nous pourrions compliquer le modèle utilisé pour le calcul effectué ci-dessus. Malheureusement celui-ci deviendrait impossible. Comme pour une usine, le calcul d'un ordonnancement optimal n'est pas réalisable. En poussant le parallélisme, nous pourrions imaginer d'utiliser un MRP pour ordonnancer l'exécution des requêtes. Mais contrairement à une ligne de production, la date de fin de traitement souhaitée pour une requête est inconnue dans le cas général. La plupart du temps, on souhaitera que la requête soit traitée au plus tôt: l'utilisateur d'un distributeur de billets de banque ne fixe pas de délai de "livraison", il veut ses billets tout de suite. L'inutilisabilité d'un MRP, tient, à mon avis, à la forte dynamique des systèmes informatiques.

L'ordonnancement étant difficile à utiliser, il faut nous intéresser à des méthodes plus « dynamiques », permettant une optimisation sans connaissance *a priori* des tâches à effectuer par le système. La méthode OPT possède ces caractéristiques.

3. Intérêt d'une approche OPT

3.1 *Justification d'une telle approche*

L'approche OPT présente un intérêt tout particulier pour un système informatique où les goulots sont souvent irréductibles. En effet, il est possible, dans une usine, d'augmenter les moyens de production affectés à une tâche, en ajoutant une machine aux machines déjà chargées de cette tâche. On peut ainsi réduire un goulot. Cette méthode de réduction est très simple à appliquer pour un processus de traitement ou un réseau; malheureusement, les goulots d'un système informatique se situent souvent au niveau des bases de données.

Une base de données doit gérer la concurrence d'accès aux données, c'est à dire empêcher que deux processus accèdent simultanément à la même donnée (pour des raisons qui sortent du champ de cette étude).

Cette nécessité implique que l'accès aux données sera forcément centralisé et contrôlé, ce qui ralentit sensiblement les processus chargés de gérer ces accès.

D'autre part, une base de données est toujours placée sur un support magnétique, peu rapide d'accès.

3.2 *Conséquences*

La performance des bases de données a été étudiée de manière approfondie durant les vingt dernières années, notamment à cause des applications de gestion qui nécessitaient des bases de données à la fois volumineuses et performantes. Les algorithmes d'accès aux données ont été optimisés, et parallèlement les performances matérielles ont cru de manière exponentielle. Malheureusement la taille des données à stocker a évolué dans les mêmes proportions. Cette évolution montre bien à quel point les bases de données sont un goulot: l'augmentation de la puissance des bases a toujours été utilisée immédiatement, ce qui prouve que l'offre de puissance est en retard sur la demande.

L'optimisation des bases de données a conduit à deux innovations majeures:

- la définition de structures de données optimisées; à ce titre les bases de données relationnelles ont représenté une grande avancée. Ces bases présentent l'intérêt d'être basées sur un modèle mathématique, et d'offrir un langage de recherche standard et efficace (SQL: Standard Query Language).
- malgré l'apparition des bases de données relationnelles, les bases de données restaient sensibles au nombre d'utilisateurs simultanés, car le temps de gestion de la concurrence d'accès aux données est proportionnelle à ce nombre. Quelle que soit la base de données, le nombre d'utilisateurs simultanés ne peut rarement excéder 100, sous peine de provoquer une chute de performances catastrophique. Il existe désormais des moniteurs transactionnels, qui gèrent eux-mêmes la concurrence d'accès aux données tout en étant l'unique utilisateur de la base. Les processus voulant accéder aux données parlent alors au moniteur transactionnel. On a ainsi réduit un goulot en isolant la cause du goulot (la gestion de la concurrence), ce qui a permis une optimisation plus poussée du fonctionnement de la ressource goulot.

3.3 *Conclusions*

Le monde de l'informatique a appliqué une démarche de type OPT durant les dix dernières années au sujet des bases de données. Les bases de données étant le goulot d'étranglement des applications, on a cherché à les optimiser. Cette approche découle, dans son esprit, d'une démarche OPT.

Nous avons donc vu, d'une part l'intérêt d'approches optimisantes, telles que OPT et d'autre part l'impossibilité de réaliser un ordonnancement de manière simple dans un cas complexe. A défaut d'un ordonnancement centralisé, essayons de réaliser un ordonnancement décentralisé par une méthode à base de flux tendus.

4. Utilisation des flux tendus

Pour comprendre l'intérêt des flux tendus dans un système informatique, il convient de préciser les conséquences des flux relâchés.

- Tout d'abord, des flux non tendus génèrent des « stocks ». Ici, les stocks seront des requêtes en attente de traitement dans un processus de traitement donné. Or, dans la plupart des systèmes, il n'est pas admissible de perdre un travail déjà effectué. Les requêtes sont alors sécurisées dans la base de données, afin de les retrouver

même après une coupure de courant. L'augmentation du volume de données à stocker, dû au trop grand nombre de requêtes stockées n'est pas un problème majeur en soi. Par contre, les temps de recherche dans la base, afin de fournir à un processus la requête la plus prioritaire le concernant, vont augmenter en fonction de la taille de cette base. Ces temps de recherche occupent inutilement les processus de gestion de la base et ralentissent donc le système.

- D'autre part, contrairement à une machine outil, les capacités de production d'un processus de traitement ne sont pas intrinsèques. Il partage en effet la puissance de calcul du système avec l'ensemble des autres processus. Considérons donc le cas suivant:

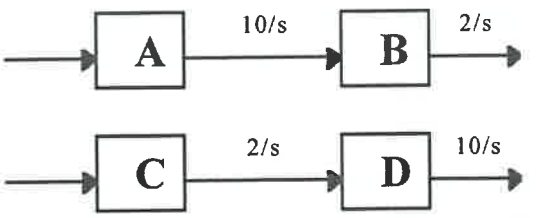


Fig. 4 : exemple de système de traitement de processus

Le processus A et le processus D traitent 10 requêtes par seconde (plus ou moins suivant la charge du système), tandis que les processus B et C n'en traitent que 2. On devine bien que:

- un stock de requêtes en attente de traitement va se créer et croître entre les processus A et B.
- le processus C n'aura pas une vitesse de traitement suffisante pour alimenter le processus D, qui va donc s'arrêter.

En appliquant une politique de flux tendus, on aurait la situation suivante:

- le processus A s'arrêterait de travailler dès qu'un stock de quelques requêtes serait généré à l'entrée de B.
- comme auparavant, le processus C aurait du mal à alimenter le processus D.
- les arrêts des processus A et D diminueraient la charge globale du système, ce qui aurait pour effet d'augmenter le rythme de traitement des processus B et C.

On conçoit donc bien qu'une politique de flux tendus aura des effets bénéfiques sur le fonctionnement du système. Etudions maintenant quelques approches « juste à temps ».

4.1 Faisabilité d'une démarche « Juste à Temps »

Une démarche JAT est adaptée à un système informatique, car elle en vérifie les conditions préliminaires. En effet, l'outil de production est fiable (les taux de fiabilité d'un système informatique sont souvent proches de 100%), et il n'y a pas d'approvisionnements, donc nous n'aurons pas le souci de les fiabiliser.

D'autre part, il sera nécessaire de rechercher les causes secondaires de défaillance. Cette politique d'amélioration du système ne peut être que bénéfique à la qualité du système. Néanmoins elle nécessitera la mise en place de traces d'exécution, affichant les principaux

paramètres du système au cours du fonctionnement. Ces traces seront utilisées pour reconstruire la série d'événements ayant conduit à une défaillance. Cette reconstruction sera d'autant plus pertinente que de nombreux événements auront été tracés dans le système. Ces traces impliquant des écritures sur disques magnétiques, ralentiront le système. Il est donc possible de mener une politique de qualité sur le long terme, mais elle sera coûteuse au niveau des performances du système à court terme (on peut en effet espérer qu'à long terme, la politique d'amélioration aura porté ses fruits et que l'on pourra alléger les traces).

Enfin, une démarche JAT impose une mobilisation de tous les intervenants. Le travail d'exploitation d'un système informatique étant relativement peu important au regard de la partie conception et développement, il conviendra d'impliquer les analystes, concepteurs et développeurs dans la démarche de qualité. Cela veut dire que la qualité du système devra être une préoccupation de premier plan, même préalablement à la réalisation du système. Cette approche est assez courante dans le développement de systèmes, et peut notamment être réalisée dans le cadre des normes ISO 9000.

4.2 Intérêt des principales démarches JAT

4.2.1 Le SMED

Le SMED n'a pas grand intérêt dans un système informatique. En effet la notion de série est la plupart du temps inexistante et les temps de changement de production sont très souvent nuls. Souvent, on préférera développer deux processus spécialisés chacun dans une tâche plutôt qu'un seul devant s'adapter à deux tâches différentes.

4.2.2 La méthode Kanban

La méthode kanban présente un certain intérêt car elle permet notamment de réduire les conséquences d'une situation telle que celle présentée en introduction de cette partie. L'application d'une méthode kanban est similaire à un cas industriel, en dehors du fait qu'elle devra être prise en compte dès la conception du système (car les modifications a posteriori sont lourdes et coûteuses).

Une méthode simple consiste à émettre avec chaque requête un kanban, qui sera renvoyé à son émetteur dès lors que la requête sera traitée. Le processus émetteur produira de nouvelles requêtes dès qu'il aura un certain nombre de kanbans en main, et s'arrêtera lorsque ce nombre deviendra trop faible. La détermination du nombre de kanbans à affecter à chaque processus est assez complexe, car la puissance de calcul étant répartie entre tous les processus, l'arrêt d'un processus accélérera les traitements en cours dans tous les autres.

On voit donc ici l'intérêt de simuler le système. De plus, un système informatique peut-être testé en grandeur réelle, et à faible coût, ce qui n'est pas souvent le cas d'une usine. Ces tests devront être réalisés durant la phase d'intégration sur site, afin de se placer au plus proche des conditions de fonctionnement. On pourra par la suite ajuster les nombres de kanbans au vu du fonctionnement réel du système.

5. Perspectives et conclusion

Au terme de cette (courte) étude, on ne peut douter de l'intérêt de l'application de la théorie des systèmes de production aux systèmes informatiques. Néanmoins ces théories sont

souvent méconnues des informaticiens, et doivent être prises en compte dès la conception des systèmes. Or le manque de temps, la limitation des moyens font que le système livré n'est jamais optimisé, et présente très souvent des dysfonctionnements. Or l'application de méthodes juste à temps, par exemple, implique une modification globale du système, ce qui n'est guère possible.

Néanmoins, les questions évoquées ici sont souvent posées et restent la plupart du temps sans réponse. Ainsi, il n'est pas rare de développer un sous-système sans connaître à l'avance les performances qu'il devra avoir pour satisfaire les performances globales du système. Afin de se prémunir contre des mauvaises surprises lors des tests en grandeur réelle, on prévoit donc à l'avance les adaptations possibles pour augmenter les performances, ou bien on optimisera un processus qui n'en avait pas besoin. L'informatique est une science encore jeune, et l'accent est pour l'instant mis plus sur la réalisation que sur les coûts. Nul doute que cette tendance est en train de s'inverser, et que la recherche d'une performance accrue à coût limité devrait conduire les chefs de projet informatique à rationaliser un peu plus les efforts portés à cet objectif.