CHAPITRE 2

COMMUNAUTÉ DE RESSOURCES

Plan

SERVICES COMMUNS
Aspect fonctionnel
Performances
Comportement
Évolution
ALLOCATION DE RESSOURCES AUX PROCESSUS
Demandes de ressources par les processus
La gestion des ressources
Stratégies globales
CHARGE D'UN SYSTÈME
PRATIQUES COURANTES

Manuel 2

Communauté de ressources

1. Services communs

1.1. Aspect fonctionnel

Un rôle important des systèmes est de fournir les services nécessaires à la communauté des processus qu'il accueille : support pour le déroulement des processus (qu'ils soient locaux ou qu'ils représentent un client distant), gestion et allocation des ressources physiques nécessaires à ces processus (hiérarchie de mémoire, processeurs, canaux de communication, organes périphériques...), contrôle du stockage et de l'échange des informations, partage et protection de ces informations, sécurité des utilisateurs contre des intrusions, et enfin enregistrement du fonctionnement (mesures, facturation, statistiques).

Ce rôle a entraîné la présentation des systèmes comme une hiérarchie de services s'appuyant sur le matériel et sur un noyau exécutif, logiciel de base s'exécutant en mode superviseur. On y trouve, par exemple, les serveurs du système d'exploitation, puis les progiciels généraux et enfin une plate-forme spécialisée pour un mode d'utilisation ou une profession. Chaque niveau de la hiérarchie implémente des services et fournit une interface vers les niveaux supérieurs, utilisateurs de ses services. L'interface du noyau s'appelle aussi les primitives systèmes. Cette présentation générale permet de spécialiser le système le plus tard et de couvrir la plupart des modes d'utilisation : ordinateur personnel, exécution des travaux en traitement différé et par lots, temps réel, interactif temps partagé ou transactionnel, réseaux, systèmes répartis. Elle permet encore de repérer les interfaces qui concernent les divers utilisateurs : client final, programmeur d'application, administrateur de réseaux, de base de données ou de systèmes, gestionnaire d'utilitaire, ingénieur système, concepteur de services, de progiciels ou de sous-système.

1.2. Performances

Cette présentation doit être complétée par des objectifs et des indicateurs quantitatifs. Le temps de réponse d'une commande mesure le délai entre l'envoi de la commande et l'arrivée de

la réponse et comprend la durée du traitement de cette commande, plus les divers délais d'attente et de propagation. Le débit d'un service dénote le nombre de commandes exécutées par unité de temps.

Les objectifs quantitatifs habituels sont de réduire le temps de réponse, d'augmenter le débit ou de respecter des contraintes temporelles comme des échéances ou l'absence de gigue. On notera que ces objectifs sont contradictoires. Selon le poids des divers objectifs qualitatifs et quantitatifs, on aboutit à des architectures logiques différentes (systèmes à partage de service et bon débit comme IBM/VM, DEC/VMS, systèmes avec gestion et protection de l'information comme Multics, Intel iAPX 432 ou IBM/AS400, systèmes multiprocessus comme Unix, Linux, systèmes temps réel comme VxWorks ou LynxOs).

1.3. Comportement

Une typologie fondée sur le comportement des systèmes face aux sollicitations externes permet de distinguer des systèmes transformationnels, interactifs ou réactifs. Dans un système transformationnel les données d'un programmes doivent être disponibles dès le début de son exécution et la date de production du résultat peut être quelconque. Dans un système interactif, les données du programme peuvent aussi être fournies par le programmeur pendant l'exécution du programme. La présence de plusieurs utilisateurs interactifs concomitants entraîne des contraintes sur le temps de réponse si on est en temps partagé ou sur la cohérence de fichiers si on est en transactionnel. Dans un système réactif, on ajoute des contraintes temporelles sur la saisie des données et la production des résultats.

1.4. Évolution

L'évolution des systèmes s'est traduite par une extraordinaire augmentation de la taille et de la complexité des programmes. Les systèmes d'exploitation sont parmi les programmes les plus gros et la concurrence d'exécution des processus qu'ils contiennent les situe dans les programmes les plus complexes qui soient. Pour lutter contre cette taille et surtout cette complexité, pour leur donner des facultés d'évolution fonctionnelle, d'ouverture vers d'autres systèmes et d'intégration de sous-systèmes rapportés, l'architecture logique des systèmes, à l'origine monolithique, a été hiérarchisée, puis a été découpée selon un schéma client serveur, ce qui a permis de la répartir. Parallèlement, le noyau exécutif, qui a l'origine contenait tous les services, a été réduit de taille et de fonctionnalités et est devenu un micro-noyau générique installé sur toutes les machines de l'architecture physique distribuée. Ces micro-noyaux coopèrent pour réaliser un système unique bien que physiquement distribué et pour gérer les processus indépendamment de leur localisation ("Single System Image").

2. Allocation de ressources aux processus

2.1. Demandes de ressources par les processus

Très généralement, toute entité, matérielle ou logicielle, qui est nécessaire à l'exécution d'un processus est considéré comme une ressource. Les ressources sont matérielles comme les processeurs, la mémoire centrale, les sous-systèmes d'entrée-sortie (processeur canal, mémoires secondaires, périphériques et réseaux). Les ressources logicielles, appelées souvent objets, comprennent, entr'autres, les procédures cataloguées, les sous-programmes de bibliothèque, les objets de synchronisation, les fichiers, les catalogues, les bases de données.

Certaines utilisations de ressources ne peuvent être effectuées que par un processus à la fois, en exclusion mutuelle ; certaines ressources peuvent être réquisitionnées à tout moment, d'autres ne sont que réutilisables après restitution dans un état correct. Dans un système, certaines ressources sont uniques alors que d'autres sont banalisées dans un lot. D'autres enfin, comme le temps ou des signaux, sont consommables et ne peuvent être rendues.

Deux aspects marquent l'originalité de l'allocation des ressources aux processus.

Le premier est que le déroulement d'un processus se fait selon le modèle architectural de Von Neumann avec un processeur et une mémoire centrale destinée à contenir tout autant le programme enregistré que les données ; ceci implique qu'un processus doit avoir conjointement de la mémoire et un processeur pour pouvoir progresser.

Le second aspect est que les demandes sont complètement imprévisibles tant pour leur date d'occurrence, que leur quantité ou la durée de leur utilisation

Pour faciliter le partage des ressources matérielles, on fait le plus possible abstraction de ces ressources en virtualisant la représentation des processus, en particulier en utilisant un adressage logique des composants du programme. Par ailleurs la réentrance du code et l'attribution d'une pile à chaque processus simplifient le partage des ressources logicielles.

2.2. La gestion des ressources

Les clients d'un gestionnaire de ressources sont des processus ou des groupes de processus ; l'accès à une ressource allouée ne doit être possible que pour l'allocataire ; et celui-ci doit en respecter les modes d'utilisation. La gestion ne doit pas mettre en danger la fiabilité du système : les ressources matérielles rendues ne doivent pas garder de trace de leur utilisation, l'allocation dynamique ne doit pas mener le système en interblocage, le partage doit être équitable, les erreurs ne doivent pas se propager par le biais des ressources partagées,...

Dans les systèmes informatiques, habituellement, les ressources sont allouées dynamiquement , celles qui ont été allouées sont capitalisées avec les nouvelles et les processus demandeurs de ressource se mettent en attente tant qu'ils ne sont pas servis.

Les processus d'un système ont un comportement inconnu a priori, et partant, les demandes de ressources sont aléatoires. L'allocateur des ressources est capable de recevoir des demandes à tout moment, de ne les honorer que si le service reste fiable, de mettre les demandeurs en attente ou de leur refuser le service. Les clients bloqués sont gérés selon une politique de file d'attente, et celle-ci peut, lorsque cela est possible, chercher à optimiser le service de la ressource. La durée d'allocation peut être laissée à la discrétion du client ou être limitée ; de toute façon, elle est sous surveillance car une ressource partagée ne peut être allouée indéfiniment.

L'allocateur gère les ressources, les repère et surveille leur utilisation ; il gère la file d'attente des clients ; il contrôle la fiabilité du service.

2.3. Stratégies globales

L'action des divers allocateurs est réglée pour fournir des performances globales. Selon les systèmes, on vise à augmenter le débit global des travaux réalisés ou à réduire le temps de réponse moyen.

L'allocation par étapes selon un ordre fixe permet de limiter le nombre maximum des processus en attente dans une file et contribue à améliorer les performances.

La fiabilité du fonctionnement contient l'absence d'interblocage et, pour les systèmes temps réel, l'absence de fautes temporelles par dépassement d'échéance. L'interblocage est un problème structurel qui ne peut être détecté ou évité que globalement.

Ces stratégies peuvent être modélisées par des files d'attente lorsqu'on étudie les performances et par des réseaux de Petri lorsqu'on veut prouver l'absence d'interblocage.

3. Charge d'un système

Pour adapter la gestion des ressources au caractère imprévisible des demandes, il est nécessaire d'en estimer quelques lois de comportement, que ce soit hors ligne ou en cours de fonctionnement, et de tenter de caractériser ainsi la charge sur le système.

Les mesures permettent de calculer des propriétés statistiques, comme la moyenne, l'écart type, et d'approcher le comportement passé des demandeurs par des lois de probabilités pour les quantités, les durées ou les intervalles entre demandes successives. Dans certains cas, on dispose des charges maximales rencontrées, ou d'un ou plusieurs échantillons de comportement nominal type ("benchmark").

La demande peut aussi être annoncée au système par l'utilisateur, qui en donne une moyenne estimée ou une valeur maximale qu'il s'engage à ne pas dépasser. Ainsi, dans une application temps réel, si l'analyse détaillée préalable du programme permet de connaître la durée d'exécution des tâches, on sait en tirer parti pour déterminer un ordonnancement fiable.

Les mesures en cours de fonctionnement permettent d'identifier la charge courante et de prédire la charge instantanée future, lorsque les comportements sont stationnaires sur le plan statistique. Elles servent alors à commander la régulation de l'allocation des ressources.

Historiquement, c'est l'analyse du comportement de l'utilisateur devant une console et la mesure de la répartition des temps de calcul des travaux interactifs et des travaux par lots, qui ont permis de déterminer le temps de réponse et de régler la valeur du quantum des premiers systèmes interactifs en temps partagé. Ce sont les mesures des suites de référence d'un processus en mémoire virtuelle paginée et les expériences de localité, qui ont permis d'améliorer la gestion de mémoire dans les systèmes de pagination à la demande.

4. Pratiques courantes

Certains systèmes, en particulier la plupart des systèmes pour ordinateurs personnels, ignorent le problème et pratiquent la politique de l'autruche. Les demandes sont servies comme elles viennent. C'est simple. Mais les temps de réponse sont très variables, dispersés, quelquefois inacceptables quand certains processus dépassent les échéances temporelles tolérables. Parfois les ressources ne sont pas disponibles au bon moment ou, pire encore, le système se retrouve en interblocage et tombe en panne. Seule solution, il faut relancer le système. Pour donner une vision moins péjorative, on donne des priorités aux clients et on essaie d'en tenir compte au mieux. On appelle ce trompe l'oeil politique du meilleur (sic!) effort ("best effort")

Dans les systèmes plus sérieux, on implante des stratégies de régulation avec des objectifs de performance ou de fiabilité. Dans ce dernier cas, les stratégies peuvent produire des refus de servir des ressources ou des retards de service. Ces retards permettent de limiter la concurrence et éviter l'interblocage ou apporter plus d'équité entre clients.

Dans certains cas, une sorte de contrat est passé entre l'utilisateur et le système, et si l'utilisateur sort des annonces du contrat, soit il n'est plus servi, soit il est pénalisé. Cette approche par contrat se rencontre dans l'annonce associée à un algorithme de prévention de l'interblocage, appelé algorithme du banquier, ou quand on essaie de garantir des qualités de service (un taux moyen de service d'unité centrale, un débit sur le réseau, un minimum de mémoire à tout instant,...).