Cours Systèmes (Distribués) Répartis

F. Baude (baude@unice.fr), Maîtrise MIAGE 2003-2004

1- Introduction

- **objectifs**
- **concepts matériels**
- **concepts logiciels**
- **bases de la conception des systèmes distribués**
- **des systèmes d'exploitation aux applications**

2- Communication dans les systèmes distribués

- **couches de protocoles**
- **modèle client-serveur**
- **appels de procédures à distance**
- **communication de groupe**
- **Quelques approches pratiques**

Plan du cours (suite)

3- Synchronisation dans les systèmes distribués

- **gestion du temps: synchronisation d'horloge**
- **élection**
- **exclusion-mutuelle**
- **transactions atomiques**
- **interblocage dans les systèmes distribués**

4- Etude de cas (TP) -- fait plus tard et dans d'autres cours !

- **Service web et programmation Client/serveur sur web**
- **Modèle CORBA et systèmes à objets distribués (? Non !)**
- **Modèles client/serveur et à objets en Java : rmi, servlet/JSP, applet**
- **Exemples d'autres applications ou services système : SG fichiers répartis, SGBD, serveurs d'applications tels EJB, etc...**

1- Introduction
1.1- Objectifs, Avantages/Inconvénients

SYSTÈME Expl. DISTRIBUÉ = SYSTÈME POSSÈDANT PLUSIEURS PROCESSEURS COOPÉRANTS

OBJECTIFS

- ***Coût*** **: plusieurs processeurs à bas prix**
- ***Puissance de calcul*** **et de** ***stockage*** **: aucune machine centralisée ne peut rivaliser**
- ***Performance (accélération)*** **: via du calcul parallèle**
- ***Adaptation*** **: à des classes d'applications réelles naturellement distribuées**
- ***Fiabilité*** **: résistance aux pannes logicielles ou matérielles**
- ***Extensibilité*** **: croissance progressive selon le besoin**

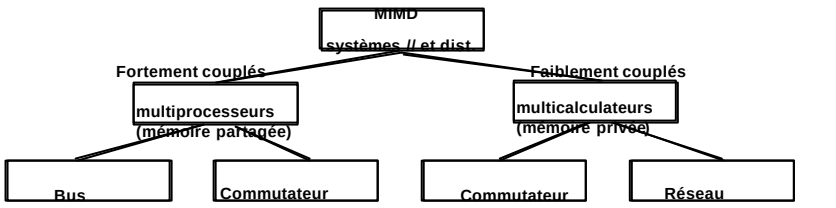
Avantages	Inconvénients
- partage de données - partage de périphériques - communication - souplesse (politiques de placement)	- logiciels : moins de logiciels disponibles - réseaux : saturation et délais - sécurité : piratage

Page 3

1- Introduction
1.2- Concepts matériels

Taxonomie de Flynn (1972)
- nombre des flux d'instructions
- nombre des flux de données

SISD : un seul flux d'instruction et un seul flux de données (ordinateurs centralisés)
SIMD : un seul flux d'instruction et de multiples flux de données (machines //, vectorielles)
MISD : multiples flux d'instruction et un seul flux de données (pas de machine réelle)
MIMD : multiples flux d'instruction et de multiples flux de données



Sequent Encore
Sun Enterprise serveurs

RP3 et SPx d'IBM

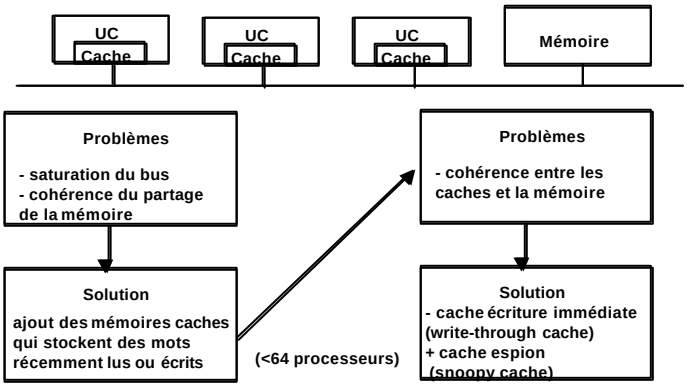
Transputer
Machines parallèles à mémoire répartie

Stations de travail sur LAN
Grappe de PCs sur un Etherswitch, ou un réseau Myrinet
Grilles sur WAN

Page 4

1- Introduction
1.2- Concepts matériels

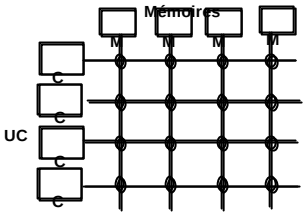
1.2.1- Multiprocesseurs à bus



Page 5

1- Introduction
1.2- Concepts matériels

1.2.2- Multiprocesseurs commutés



Technique de matrice de commutation (Crossbar Switch)
- les UC et mémoires sont reliées par une matrice de commutation
- à chaque intersection, le noeud de commutation (crosspoint switch) peut-être ouvert ou fermé
- quand une UC veut accéder à un module de mémoire elle ferme temporairement le noeud de commutation correspondant
- si plusieurs UC veulent accéder au même module, une file d'attente est nécessaire

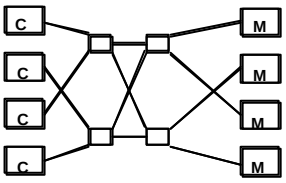
Inconvénient
le nombre des noeuds de commutation nécessaires : il faut n^2 de noeuds de commutation pour relier n UC aux n modules de mémoire

⇒ réseaux d'interconnexion minimisant le nombre de noeuds ?

Page 6

1- Introduction
1.2- Concepts matériels

1.2.2- Multiprocesseurs commutés



Principe d'un réseau multi-étages: ex. omega

- utilisation des commutateurs 2x2 : 2 entrées et 2 sorties
- chaque commutateur peut relier n'importe quelle entrée à n'importe quelle sortie
- pour relier n UC à n mémoires, il nécessite $\log_2 n$ étages dont chacun contient n/2 commutateurs 2x2
- le nombre nécessaire de commutateurs est : $n \times \log_2 n$

Inconvénient : Temps de propagation

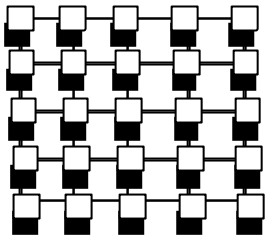
- Si n= 1024, on a besoin 10 étages
- Avec les UC de 50Mhz, le cycle de calcul est de 20ns
- une requête mémoire traverse 20 étages (aller/retour) en 20 ns si le temps de commutation est de 1ns
- On doit avoir 10 240 commutateurs à 1ns !!!

CONCLUSION

construire un gros multiprocesseurs fortement couplé à mémoire partagée est difficile et coûteux

1- Introduction
1.2- Concepts matériels

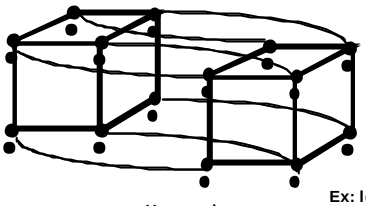
1.2.3- Multicalculateurs commutés



Topologie en Grille de 2 dimensions

- câblage simple
- le chemin plus long croit en racine carrée du nombre d'UC
- la brique de base est tjs la même

Ex: Machines ad-hoc sur Transputer, T3x (Cray): grille torique
Connection Machine 2 et 5 (Fat tree)



Hypercube (4 dimensions)

Ex: le n-cube -> 8192 UC

- Un hypercube est un cube à n dimensions
- 4 dimensions : 2 cubes de 3 dimensions avec les sommets homologues reliés
- 5 dimensions : 2 hypercubes de 4 dimensions avec les sommets homologues reliés, etc...
- la complexité du câblage croit en $\log_2$ du nombre UC
- le chemin le plus long croit en $\log_2$ du nombre UC

Absence de Mem. Centrale commune

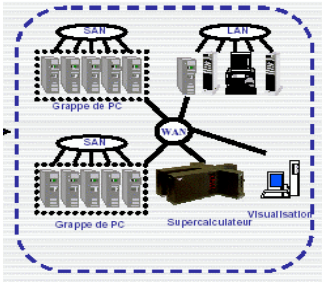
=> Trouver un autre moyen pour partager de l'information = logiciel

- Mémoire Virtuellement Partagée -> problématique NUMA
- ou, mode de programmation fondé uniquement sur l'échange de données

1- Introduction
1.2- Concepts matériels

1.2.4- Multicalculateurs faiblement interconnectés au dessus d' un réseau Internet (IP) : meta calculateur

Grappes (cluster)



Grilles (cf. "Power Grid")

- technologie réseau rapide (giga bits Ethernet, Myrinet, SCI)
- un seul frontal (console) : unique point d'accès au cluster -- soumission de jobs (LSF,PBS,...)
=> tous les noeuds de la grappe sont dans le même domaine d'administration
Ex: Clusters de type "beowulf" (PC du commerce + Linux + Ethernet)
Clusters d'Intel, d'AMD Athlon, ...

Page 9

- de stations de travail (desktop grids) et/ou de PC "volontaires"
- de grappes
=> toutes combinaisons possibles
- réseau sous-jacent si possible pas trop lent (ex WAN haut débit comme Renater 3 -- Gigabits/sec)
=> difficile problème provenant de la non unicité du domaine d'administration
(authentification -- compte invité ; nombreux points d'entrée effectifs; soumission de jobs sur la grille entière : par découpage en sous-ens de jobs vers +eurs noeuds de la grille)
Le futur : Agréger des BDs -- ex BD de proteines -> cancer

1- Introduction
1.3- Concepts logiciels

1.3.1- Classification

Le matériel est important, mais le logiciel de base (SE ou plus généralement Syst. Operatoire) l'est plus encore !

La classification est, par nature, imprécise et floue :

systèmes d'exploitation faiblement couplés	systèmes d'exploitation fortement couplés
- réseaux d'ordinateurs indépendants (et alors les ordinateurs indépendants avec réseau défaillant continuent +/- à fonctionner) (Ex les 15000 PCs du cluster Google ou toute archi. géographiquement repartie)	- multiprocesseurs faits pour exécuter un seul programme en // (jeu d'échecs, calcul scientifique) - machines dédiées (ex: // intra –plutot que inter-- requete sur BDs -- cf Oracle //, Informix, etc)

multiprocesseurs SE faiblement couplé	multiprocesseurs SE fortement couplé	Systèmes d'exploitation distribués Systèmes d'exploitation réseaux Systèmes multiprocesseurs à temps partagé
multicalculateurs SE faiblement couplé	multicalculateurs SE fortement couplé	

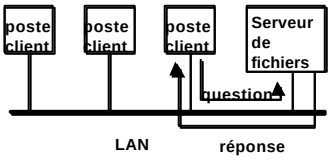
Page 10

1- Introduction
1.3- Concepts logiciels

1.3.2- Systèmes d'exploitation réseau et NFS

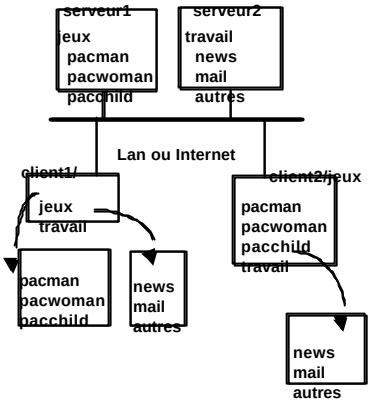
La combinaison la plus fréquente : matériels et logiciels faiblement couplés

Connexion point-à-point		système d'exploitation réseau
- postes avec SE indépendant - partage des imprimantes - accès à un autre poste en mode terminal		- postes indépendants + serveurs de fichiers - systèmes de fichiers partagés (NFS) - gestion centrale des utilisateurs (CD, NIS)
Réseaux locaux - stations NT - windows workgroups	Internet FTP, TELNET, HTTP, RLOGIN..	Novell, LanManager Unix+NFS, NTserver services



1- Introduction
1.3- Concepts logiciels

Architecture NFS (Sun)



Serveurs

- exporter des répertoires (/etc/exports)
- gérer les accés aux répertoires exportés pour les clients (locaux ou à distance)
- protection des fichiers, cohérence
- transmettre les données aux clients (réponses)
- les clients et les serveurs peuvent partager un même système de fichiers

Clients

- monter des répertoires exportés sur sa station
- utiliser ces répertoires comme (ou presque) ceux locaux
- plusieurs clients peuvent partager les mêmes répertoires (fichiers partagés)
- toute machine peut être à la fois client et serveur
- systèmes d'exploitation locaux peuvent être <>

Protocoles NFS

Un protocole est un ensemble de requêtes envoyées par les clients aux serveurs auxquelles correspondent les réponses des serveurs aux clients

1er protocole : Protocole de montage

1- Le client envoie une requête contenant le chemin d'accès à un répertoire du serveur et une demande d'autorisation de le monter dans son arborescence.

2- Si le chemin est correct et le répertoire est exportable, le serveur revoie au client une clé de fichier (file handle) concernant ce répertoire. Cette clé comporte des champs qui identifient uniquement le type de Système de Fichiers, le disque, le n° de i-noeud du répertoire et les informations sur les protections.

3- Classiquement, le client a un fichier Shell script /etc/init.d/netfs contenant entre autre les commandes de montage. Ce Shell script sera exécuté automatiquement au lancement du système

4- Automontage (Unix de Sun) : le montage se fait automatiquement à la 1ere ouverture d'un fichier à distance, le client contacte les serveurs et monte le répertoire contenant ce fichier

Page 13

2ème protocole : Protocole d'accès

1- Le client envoie au serveur des requêtes de manipulation des répertoires ou des fichiers

2- La plupart des appels système UNIX sont supportés par NFS, à l'exception de OPEN et CLOSE.
L'omission de ces opérations permet aux serveurs de ne pas gérer les fichiers ouverts (serveur sans état (stateless)). Chaque requête d'accès se suffit à elle-même. L'appel READ contient la clé du fichier, la position courante et le nombre d'octets à lire. On utilise LOOKUP à la place de OPEN.
L'inconvénient principal est l'absence de contrôle de cohérence : On ne peut pas verrouiller le fichier (car il est toujours à l'état fermé), donc le partage d'accès peut introduire des incohérences (=> lock manager).
De plus, pour des raisons de performance, les clients utilisent des caches => MAJ incohérentes
Ex: Append difficile !

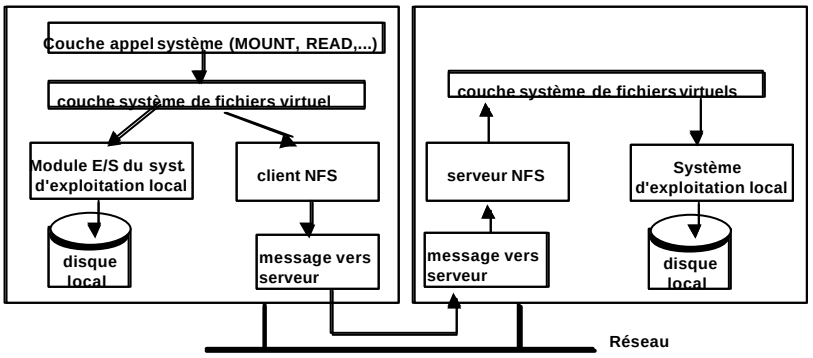
Rem- Il existe sous UNIX système V (uniquement), le Remote File System (RFS) qui demande que le fichier soit ouvert avant la lecture. Le serveur doit gérer dans ce cas un descripteur de fichier pour tout fichier ouvert à distance (statefull, mais sémantique du SGF Unix respectée)

Page 14

Mode de fonctionnement de NFS

Point de vue d'une application Cliente, tournant sur une machine dite "cliente"

Machine dite "Serveur"



Page 15

1- Introduction

1.3- Concepts logiciels

1.3.3- Systèmes distribués

Un système distribué est un système qui s'exécute sur un ensemble de machines sans mémoire partagée, mais que l'utilisateur voit comme une seule et unique machine au point que la défaillance d'une machine dont l'utilisateur ignore jusqu'à l'existence peut rendre sa propre machine inutilisable…

Caractéristiques

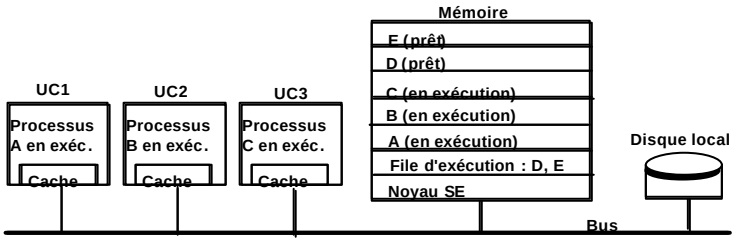
- possédant un mécanisme de communication interprocessus unique et global permettant le dialogue entre deux processus quelconques
- possédant un système de protection unique et global
- possédant un mécanisme de gestion de processus unique
- possédant un ensemble d'appels système unique, disponible sur toutes les machines
- possédant un noyau de système d'exploitation identique implanté sur toutes les machines
- possédant un mécanisme de gestion de mémoire identique

Page 16

1- Introduction

1.3- Concepts logiciels

1.3.4- Systèmes multiprocesseurs à temps partagé



- Un ensemble de processeurs "symétriques"
- Une mémoire commune et un espace disque commun
- Tous les programmes sont stockés dans la mémoire partagée
- Une file d'attente unique des processus exécutables se trouvent dans la mémoire partagée
- L'ordonnanceur travaille en section critique pour éviter que deux UC ne choisissent le même processus à exécuter
- L'exclusion mutuelle peut être réalisée en utilisant des sémaphores, moniteurs, ...

1- Introduction

1.3- Concepts logiciels

1.3.5- Tableau de synthèse sur la classification des Systèmes d'exploitation

Questions	S.E.Res.	S.E.Dist.	S.E.MemP.
Ressemble t-il à un monoprocesseur virtuel ?	Non	Oui	Oui
Doit-on avoir le même système d'exploitation (local) ?	Non	Oui	Oui
Combien y a-t-il de copies du système d'exploitation ?	N	N	1
Quel est le mode de communication d'info naturel ?	Fichiers partagés +messages	Messages	Mémoire partagée
Existe-il une seule file d'attente des exécutables ?	Non	Non mais coopération	Oui

1- Introduction

1.4- Critères de Conception d'un S.E.D. ou d'un Syst. opératoire pour une architecture distribuée

1.4.1- Transparence

Même mode d'utilisation que celui d'un système centralisé à temps partagé : au niveau d'un l'utilisateur et au niveau d'un programme (appels au système)

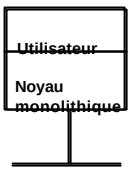
Types de transparence	Signification
Transparence à l'emplacement	L'utilisateur ne connaît pas où sont situées les ressources
Transparence à la migration	Les ressources peuvent être déplacées sans modification de leur nom
Transparence à la duplication	L'utilisateur ne connaît pas le nombre de copies existantes
Transparence à la concurrence	Plusieurs utilisateurs peuvent partager en même temps les mêmes ressources
Transparence au parallélisme	Des tâches peuvent s'exécuter en parallèle sans que les utilisateurs le sachent

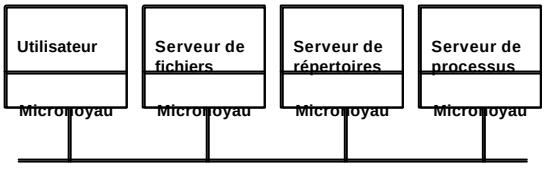
1- Introduction

1.4- Critères de Conception d'un S.E.D. ou d'un Syst. opératoire pour une architecture distribuée

1.4.2- Souplesse

La facilité de modification, de configuration et d'extension





Bus / réseaux locaux

École Monolithique

- 1ers Unix en particulier
- Linux (meme si modules)

École Micronoyau

- WindowsNT ... et encore, pas pour tout !
- Mach (Unix/Posix)

=> idee intéressante qui permet de bâtir un système operatoire pour grappe ou grille, sur des systemes d'exploitation usuels (par ex. gestion transparente des soumissions de tâches utilisant des sondes de performance.)

1- Introduction

1.4- Critères de Conception d'un S.E.D. ou d'un Syst. opératoire pour une architecture distribuée

1.4.3- Fiabilité

Disponibilité

La disponibilité est la fraction de temps pendant laquelle le système est utilisable :

- **limiter le nombre des composants critiques**
- **dupliquer les parties clés des composants logiciels et matériels (redondance)**
- **mais implique de savoir maintenir la cohérence des copies**

Sécurité

Les ressources doivent être protégées contre des utilisations abusives et malveillantes. En particulier le problème de piratage des données sur le réseau de communication

Tolérance aux pannes

Le système doit être conçu pour masquer les pannes aux utilisateurs. La panne de certains serveurs (ou leur réintégration dans le système après la réparation) ne doit pas perturber l'utilisation du système en terme de fonctionnalité (NFS sans état par exemple)

1- Introduction

1.4- Critères de Conception d'un S.E.D. ou d'un Syst. opératoire pour une architecture distribuée

1.4.4- Performances

Critères

- **temps de réponse**
- **débit (nombre de travaux par heure)**
- **taux d'utilisation du système**
- **pourcentage utilisé de la bande de passante du réseau**

Problèmes

- **La communication est en général assez lente dans les systèmes distribués par rapport accès fichier**
- **Le mécanisme de tolérance aux pannes requiert beaucoup d'opérations "inutiles" si pas de panne !**

Solutions

Minimiser les échanges de message par exemple en faisant en sorte de

- **maximiser des granules grosses (gros grains) etéviter des grains fins pour les calculs à distance**
- **réduire le champ d'application des mécanismes de reprises sur pannes (car cela engendre des transmissions d'information, et donc, les réserver aux parties critiques)**

1- Introduction

1.4- Critères de Conception d'un S.E.D. ou d'un Syst. opératoire pour une architecture distribuée

1.4.5- Dimensionnement

Goulots d'étranglement potentiels

Concepts	Exemple
Composants centralisés	Un seul serveur de courrier pour tous les utilisateurs
Tables centralisées	Un seul annuaire en ligne
Algorithmes centralisés	Avoir un routage qui nécessite une connaissance totale du réseau

- Caractéristiques des algorithmes distribués :

  - aucun processus n'a une information complète sur l'état du système
  - les processus prennent des décisions à partir des seules informations locales disponibles
  - la panne d'un processus n'empêche pas l'algorithme de fonctionner
  - on ne fait pas l'hypothèse de l'existence d'une horloge physique globale entre les machines sur lesquelles tournent les processus mettant en oeuvre l'algo.

1- Introduction

1.5- Des Systèmes d'exploitation réseau ou distribués (répartis) aux Applications distribuées (réparties)

- La transition est naturelle. Des mêmes exigences, des mêmes problèmes, le même genre de solutions !
- Par exemple: pour un SED, il faut des mécanismes de communication logiciels pour faire coopérer :

→ les noyaux de SE locaux,

→ les clients avec les serveurs offrant des services système (ex, clients et serveurs NFS),

→ les différents serveurs pour des soucis de tolérance aux pannes, etc

=> on utilise couramment des sockets ou des RPC entre processus

- Pour une Appli Distribuée, il faut des mécanismes de communication similaires entre

→ les exécutifs des langages (ex entre JVMs, entre JVM et serveur de noms/localisation)

→ entre les entités programmées dans ces langages (ex, entre Objets Java)

=> on utilise couramment des envois de messages ou des RMI, qui seront construits au-dessus de ceux offerts par le SED

- On peut faire le même parallèle par ex avec un service de mémoire partagée offert par le SE ou par le niveau applicatif (ex, JavaSpaces); avec un service de communication collective (ex, JMS)
- Le niveau applicatif permet souvent de pallier aux insuffisances du SE ! (avec en plus, plus de portabilité)

1- Introduction

1.6- Quelques questions

1) Donner deux avantages des systèmes distribués sur les systèmes centralisés

2) Un multiprocesseur à bus utilise un cache espion pour gérer une mémoire cohérente. Peut-on alors utiliser des sémaphores ?

3) Un multicalculateur comprenant 256 UC est organisé en grille de 16x16. Quel est, dans le pire des cas, le délai de transmission d'un message (exprimer en nombre de passages d'une UC à la suivante) ?

4) Considérons maintenant un hypercube de 256 UC. Quel est alors le délai de transmission dans le pire de cas ?

5) Quels avantages/inconvénients y a-t-il à ce que les serveurs NFS soient conçus pour être sans état ?

6) NFS permet à une machine d'être à la fois client et serveur. Est-ce utile ? Pourquoi?

7) Quelles sont les tâches essentielles d'un micronoyau ? Quels sont les avantages des micronoyaux sur les noyaux monolithiques ?

8) Pourquoi un cache 'write-through' est insuffisant dans le cas d'une architecture à mémoire partagée, où chaque UC a un cache local ?

1- Introduction

1.6- Quelques pistes de réponses

1) Donner deux avantages des systèmes distribués sur les systèmes centralisés

Redondance matérielle => tolérance augmentée ; Partage de la charge selon fonctionnalités (ex, mail, news, home dirs)

2) Un multiprocesseur à bus utilise un cache espion pour gérer une mémoire cohérente. Peut-on alors utiliser des sémaphores ?

Oui, car si on 'prend' ou on 'met' une bille du sémaphore, c'est une écriture mémoire que le cache espion verra donc => invalidation de cache

3) Un multicalculateur comprenant 256 UC est organisé en grille de 16x16. Quel est, dans le pire des cas, le délai de transmission d'un message (exprimer en nombre de passages d'une UC à la suivante) ?

Le diamètre d'une grille est la racine carrée du nombre d'UC, donc ici de 16

4) Considérons maintenant un hypercube de 256 UC. Quel est alors le délai de transmission dans le pire de cas ?

Le diamètre d'un hypercube est le logarithme en base 2 du nombre d'Ucs, soit 8.

5) Quels avantages/inconvénients y a-t-il à ce que les serveurs NFS soient conçus pour être sans état ?

1 Avantage: après panne, la reprise se fait sans besoin de restauration d' états des fichiers qui étaient en cours d'utilisation. 1 Inconvénient: la sémantique n'est pas parfaitement identique au cas centralisé Unix.

6) NFS permet à une machine d'être à la fois client et serveur. Est-ce utile ? Pourquoi?

Oui, parfaitement utile: en tant que client, par exemple importer la partition de mails; en tant que serveur, par exemple, exporter /usr/ si par exemple, on y a installé un logiciel particulier (mais pas installé sur les autres machines du réseau)

7) Quelles sont les tâches essentielles d'un micronoyau? Quels sont les avantages des micronoyaux sur les noyaux monolithiques ?

Gérer les interruptions, matérielles ou logicielles; ordonnancer l'exécution des processus; bref, faire le lien entre le matériel et le logiciel. Les noyaux monolithiques sont difficilement extensibles et maintenables. Avec un micronoyau on peut déporter en mode utilisateur diverses fonctionalités que l'on trouve souvent au sein de noyaux monolithiques (ex, authentifier les connexions des utilisateurs). Avec un micro-noyau, il est donc plus simple de mettre à jour ceci.

8) Pourquoi un cache 'write-through' est insuffisant dans le cas d'une architecture à mémoire partagée, où chaque UC a un cache local ?

Car si les Ucs ont dans leur cache local une variable qui est modifiée directement en mémoire centrale, ils n'auront pas pour autant la dernière valeur.