

THESE
de
DOCTORAT de L'UNIVERSITE PARIS 6

Spécialité
INFORMATIQUE

présentée par
Mario PAPAGEORGIU

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE PARIS 6

Sujet de la Thèse :
LES SYSTEMES DE GESTION DE FICHIERS REPARTIS

Soutenu le 22 Janvier 1988 devant le jury composé de :

MM.	G. PUJOLLE	Président
	C. BETOURNE	Rapporteurs
	M. SHAPIRO	
	C. GIRAULT	Examineurs
	M. GUILLEMONT	
	J. ROGADO	

RESUME

Dans un système de gestion répartie de fichiers transparent, la répartition des fichiers doit être cachée à l'utilisateur, qui a alors l'impression de travailler sur un système intégré, logiquement unique : l'arborescence de désignation est globale, unique et uniforme et englobe tous les fichiers du système ; l'accès à un fichier local ou distant se fait de la même façon (transparence d'accès) avec des noms globaux (transparence de nom) et indépendants de la localisation (transparence de localisation).

Cette thèse étudie d'abord les différents concepts et approches de réalisation des systèmes de fichiers répartis existants. Elle décrit ensuite notre solution, mise en oeuvre dans le système réparti Chorus, offrant une interface compatible avec celle du système Unix, et permettant les accès répartis transparents aux fichiers. La transparence dans notre système de fichiers est obtenue grâce à l'introduction de noeuds de type **lien symbolique** et de noeuds de type **porte**. Ces noeuds font partie de l'arborescence du système de fichiers ; ils sont désignés de la même façon que les fichiers classiques mais remplissent des fonctions particulières, déclenchant des actions spécifiques quand ils sont rencontrés pendant une analyse de nom. Les noeuds de type lien symbolique permettent l'interconnexion arbitraire des arborescences des serveurs de fichiers, alors que les noeuds de type porte fournissent un moyen de redirection de requêtes entre serveurs. La localisation d'un fichier fait partie de l'analyse de son nom, et s'exécute de façon répartie : chaque serveur analyse la partie du nom qu'il connaît, et redirige le reste vers un autre serveur. Bien que les noeuds de type lien symbolique et les noeuds de type porte puissent être utilisés séparément, ils sont conçus pour fonctionner ensemble, et offrent une arborescence-réseau logique unique et permettent les accès répartis transparents des usagers aux fichiers. Ils sont aussi utilisés pour la mise en oeuvre d'un mécanisme de mise en cache de fichiers distants.

MOTS CLES

Unix réparti
fichiers répartis
désignation globale
localisation

Chorus
répartition transparente
analyse répartie de noms
mise en cache

ABSTRACT

In a transparent distributed file system, the physical distribution of files should be hidden from the user, who would then have the perception of a single, integrated system : the name tree is global, uniform and spans all the files across the network ; access is done in the same way for both local and remote files (access transparency) using location-independent (location transparency) global names (name transparency).

This thesis first examines the concepts and approaches used in existing distributed file systems. Then it presents our solution, implemented for the Chorus distributed operating system, offering a Unix-compatible interface and transparent network-wide access to files. Transparency in our file system is provided by means of two special node types, the **symbolic link** and the **port**. These nodes are registered on the name tree and are named as normal files, but have a special role and trigger specific actions when encountered during pathname analysis. A symbolic link node arbitrarily interconnects the name trees of the file servers, whereas the port node forwards requests between servers. Locating a file is part of distributed pathname analysis : each server analyzes the local part of the pathname, and forwards the rest to another server. Although the symbolic link and port nodes can be used separately, they are meant to work together, providing a single network-wide name tree and transparent access to files. They are also used in the implementation of a file cache mechanism.

KEYWORDS

distributed Unix
distributed file systems
network-wide naming
localizing

Chorus
transparent distribution
distributed pathname analysis
caching

TABLE DES MATIERES

INTRODUCTION

1. LES SYSTEMES D'EXPLOITATION REPARTIS
2. LES SYSTEMES DE GESTION DE FICHIERS REPARTIS
3. LE SYSTEME D'EXPLOITATION REPARTI CHORUS
4. SUJET ET PLAN DE LA THESE

CHAPITRE 1. Concepts de base des systèmes de gestion de fichiers répartis

INTRODUCTION

1. DEFINITIONS
2. CADRE DE CETTE ETUDE
3. OBJECTIFS D'UN SYSTEME DE FICHIERS REPARTI IDEAL
4. COUCHES DE DESIGNATION DANS UN SYSTEME REPARTI
 - 4.1 Présentation des différentes couches
 - 4.2 Les noms internes
 - 4.2.1 Les identificateurs uniques et globaux (UID)
 - 4.2.1.1 Structure des UID
 - 4.2.1.2 Avantages de l'utilisation des UID
 - 4.2.2 Les points d'accès
 - 4.2.3 Distinction entre point d'accès et UID
5. L'ARBORESCENCE DE DESIGNATION REPARTIE
 - 5.1 La super-racine ou racine réseau
 - 5.2 Le montage de volumes à distance
 - 5.3 Arborescence de désignation unique ou virtuellement centralisée
6. LA LOCALISATION DANS UN SYSTEME REPARTI
 - 6.1 Localisation dans les systèmes à désignation non transparente
 - 6.2 Localisation dans un système virtuellement centralisé
 - 6.2.1 Localisation à l'aide du nom symbolique
 - 6.2.1.1 Localisation intégrée dans le SGFR
 - 6.2.1.2 Localisation par des préfixes
 - 6.2.2 Localisation à l'aide d'identificateurs uniques et globaux (UID)
 - 6.3 Serveur de noms et serveur de localisation
7. QUELQUES AUTRES FONCTIONNALITES INTERESSANTES
 - 7.1 La duplication
 - 7.2 L'utilisation d'un cache
 - 7.3 Les transactions atomiques
 - 7.4 La tolerance aux fautes
 - 7.5 Identification des usagers et protection

CONCLUSION

TABLE DES MATIERES

CHAPITRE 2. La transparence, son utilité et ses limitations

INTRODUCTION

1. LE CONCEPT DE LA TRANSPARENCE
 - 1.1 Aspects de la transparence
 - 1.2 La transparence aux différents niveaux du système
 - 1.3 L'appel de procédure à distance (RPC)
2. LA TRANSPARENCE TOTALE EST-ELLE POSSIBLE ?
 - 2.1 Espace local de désignation
 - 2.2 L'utilisation de contextes
3. ENTRAVES A LA TRANSPARENCE
 - 3.1 Hétérogénéité
 - 3.2 Autonomie locale des machines
 - 3.3 Interconnexion par des réseaux publics

CONCLUSION

CHAPITRE 3. Présentation de solutions dans différents systèmes

INTRODUCTION

1. LA NEWCASTLE CONNECTION
 - 1.1 Identification des usagers et contrôle d'accès
 - 1.2 Implantation
2. L'HETEROGENEITE DANS LE SYSTEME NFS
 - 2.1 Objectifs et architecture
 - 2.2 Interfaces du système de fichiers
 - 2.3 Implantation
 - 2.3.1 Le protocole
 - 2.3.2 Le serveur
 - 2.3.3 Le client
3. L'ADMINISTRATION DANS ITC
 - 3.1 Objectifs et architecture
 - 3.2 Le concept de volume
 - 3.3 Opérations sur les volumes
4. LA DUPLICATION DES FICHIERS DANS LOCUS
 - 4.1 La synchronisation pour l'accès aux fichiers
 - 4.2 Identification des duplicata
 - 4.3 Extension de la table des volumes montés
5. LA MEMOIRE VIRTUELLE RESEAU DANS APOLLO
 - 5.1 Présentation
 - 5.2 Le système de stockage d'objets

5.3 La mémoire virtuelle réseau à un seul niveau

5.4 Le mécanisme de mise en cache

CONCLUSION

CHAPITRE 4. Présentation du système réparti Chorus

EVOLUTION DU SYSTEME CHORUS

1. LES TROIS CONCEPTS DE BASE DE CHORUS

1.1 Les acteurs

1.2 Les portes

1.3 Les messages

2. LE SYSTEME D'EXPLOITATION

2.1 Structure interne du système

2.2 L'interface du système

2.3 L'implantation de Chorus sur la SM90

3. LES GROUPES DE PORTES

3.1 Définition et utilité

3.2 Exemples d'utilisation

4. CONSEQUENCES DE LA COMPATIBILITE AVEC UNIX

4.1 La création d'acteurs

4.2 Héritage du contexte fichier

4.3 Héritage du contexte de communication

4.4 La porte ombilicale

CONCLUSION

CHAPITRE 5. Le système de gestion de fichiers réparti dans Chorus

INTRODUCTION

1. OBJECTIFS DU SYSTEME DE FICHIERS REPARTI DANS CHORUS

2. L'ACCES AUX FICHIERS DISTANTS

2.1 L'acteur gestionnaire de fichiers (AGF)

2.2 La bibliothèque d'interface

2.3 Identification unique d'un fichier

2.3.1 Le cas Unix

2.3.2 Le cas Chorus. Les points d'accès

2.3.3 Opérations sur un point d'accès

2.4 Le contexte fichier

2.4.1 Attachement du contexte fichier à l'acteur

2.4.2 Description du contexte fichier d'un acteur

2.4.3 L'AGF, serveur sans contexte

TABLE DES MATIERES

2.4.4 La bibliothèque d'interface et le contexte fichier

2.5 Le problème de la transparence de localisation

3. L'ARBORESCENCE DE DESIGNATION REPARTIE

3.1 Les noeuds de type porte

3.2 Les noeuds de type lien symbolique

3.3 Utilisation des noeuds de type lien symbolique et porte

3.4 Espace de désignation local et partagé. Fichiers dupliqués

3.5 Initialisation des noeuds de type lien symbolique et porte

3.6 Exemple global de fonctionnement

3.7 Désignation des autres objets

3.8 Détection de boucles dues aux liens symboliques •

3.9 Noeuds de type lien symbolique et migration de fichiers

4. IDENTIFICATION DES USAGERS ET PROTECTION

5. UTILISATION DU SGFR PAR LES USAGERS

5.1 Mobilité des usagers

5.2 Migration des fichiers vers l'utilisateur

5.3 Sauvegarde des informations

6. LES PERIPHERIQUES DE TYPE CARACTERE

6.1 Principe

6.2 Chargement d'un acteur *driver*

6.3 Protocoles à respecter

7. COMPARAISON AVEC D'AUTRES SOLUTIONS

7.1 Construction de l'arborescence de désignation

7.2 Méthodes de localisation

7.3 Niveaux d'implantation

CONCLUSION

CHAPITRE 6. Duplication et mise en cache dans un système réparti : le cas Chorus

INTRODUCTION

1. LA DUPLICATION DE FICHIERS

1.1 Différences par rapport aux bases de données

1.2 Les différentes formes de cohérence

1.3 Cohérence mutuelle entre duplicata

1.4 Conclusion sur la duplication

2. LA MISE EN CACHE DANS UN SYSTEME REPARTI

2.1 La cohérence interne

2.2 La cohérence mutuelle

2.3 Le cache de suggestions

3. LA MISE EN CACHE DANS CHORUS

3.1 Le cache de chemins d'accès

3.1.1 Le rôle d'un cache de chemins d'accès

3.1.2 Un cache de chemins d'accès ayant la forme d'un arbre

3.1.3 La cohérence du cache

3.1.4 Fonctionnement

3.1.5 Evaluation quantitative du cache

3.1.6 Conclusion sur le cache de chemins d'accès de Chorus

3.2 Le cache des fichiers distants

3.2.1 Nature de l'environnement

3.2.2 Synchronisation pour l'accès aux fichiers mis en cache

3.2.3 Désignation et stockage des fichiers cachés. Catalogues cachants

3.2.4 Désignation des copies cachées. Fichiers de type anti-cache

3.2.5 Conclusion sur la mise en cache des fichiers distants dans Chorus

CONCLUSION

REFERENCES BIBLIOGRAPHIQUES

ANNEXE 1. Liste des figures

ANNEXE 2. Brève description du système Unix

ANNEXE 3. Extrait du manuel de programmation du système Chorus