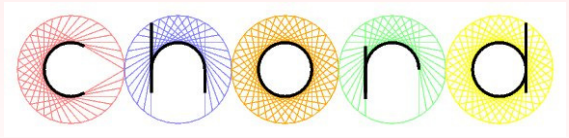
A Peer-to-Peer Replica Location Service Based on A Distributed Hash Table

Min Cai, Ann Chervenak et Martin Frank

USC Information Sciences Institute

California, U.S.A



Plan

- Introduction
- Réplique des données
- Replica Location Service
- Peer to Peer
- Chord
- Peer to Peer Replica Location Service
- Conclusion

Introduction

Grille de Donnée

↓

Réplique des Données

↓

Système de gestion des Répliques

↓

Replica Location Service

Le projet Datagrid

La Réplique des données

Définition

Une technique d'optimisation bien connue dans les systèmes distribués et les communautés des Bases de données comme un moyen pour réalisé un meilleurs temps d'accès aux données et / ou une disponibilité et une tolérance au défaut en dupliquant les données

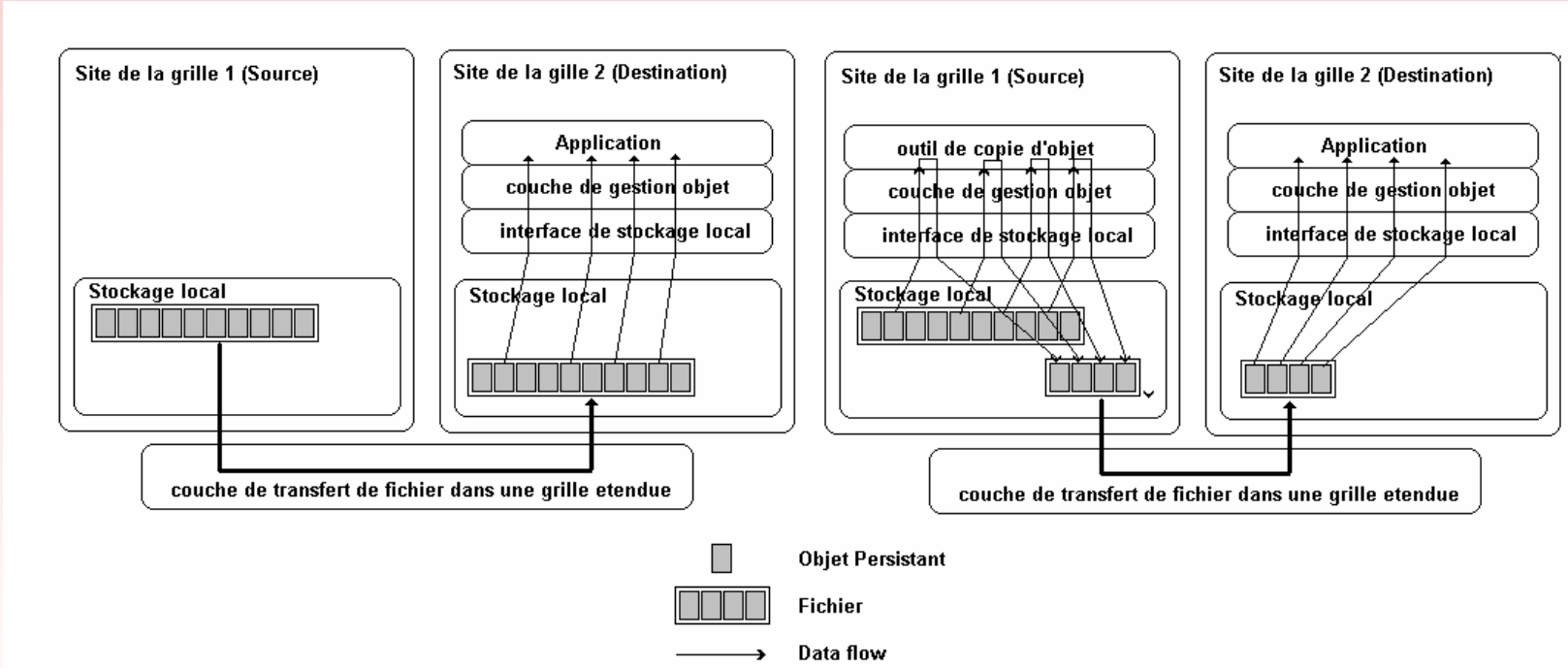
Exemple de Besoin

La Réplique des données

La Nécessité de dupliquer les données

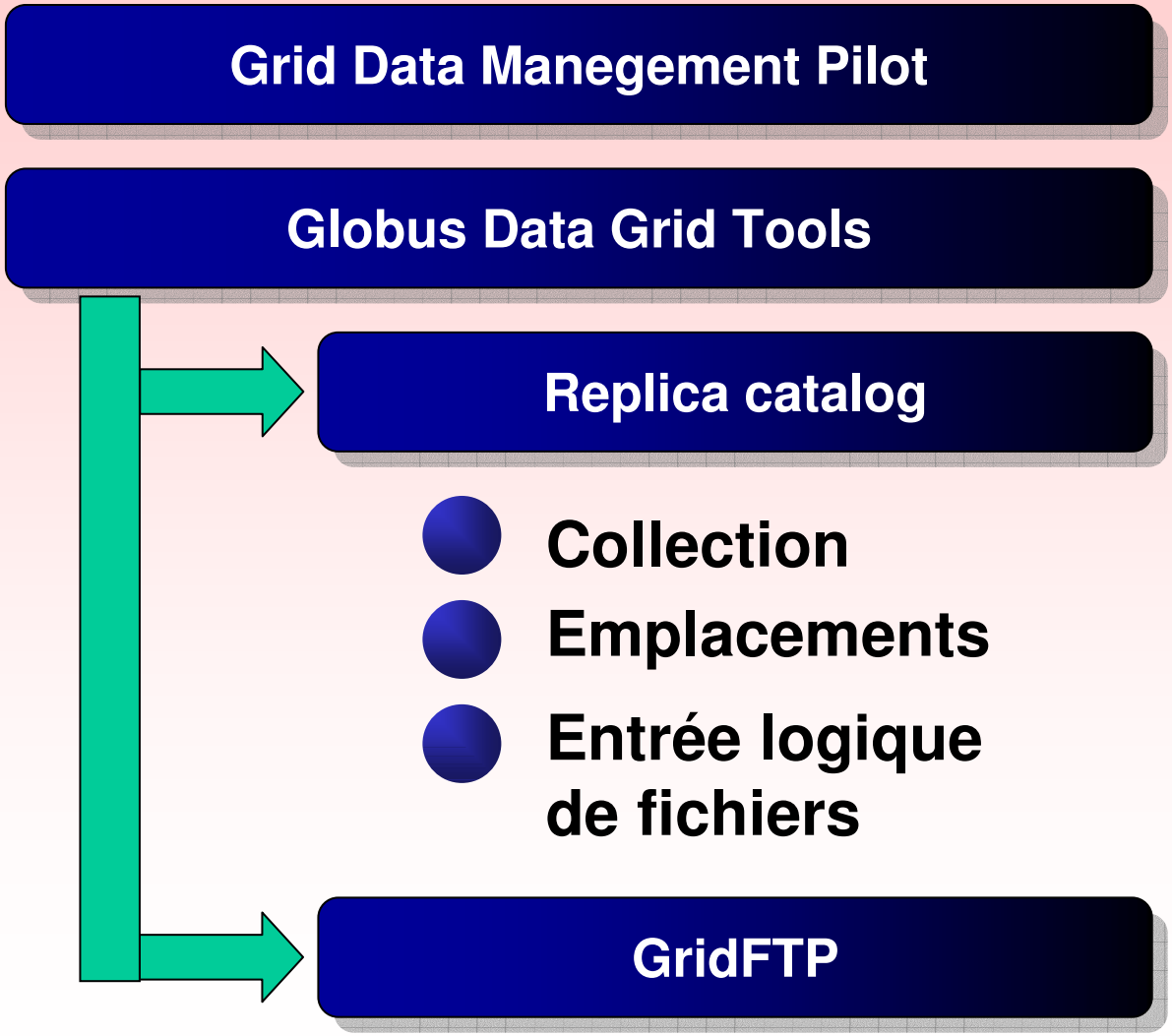
- Les objets sont Read-Only lors de la création
- Les ressources sont très distribuées
- L'accès est répétitif
- L'abstraction des données
- L'incapacité d'avoir un accès efficace a distance au objets

La Réplique des données

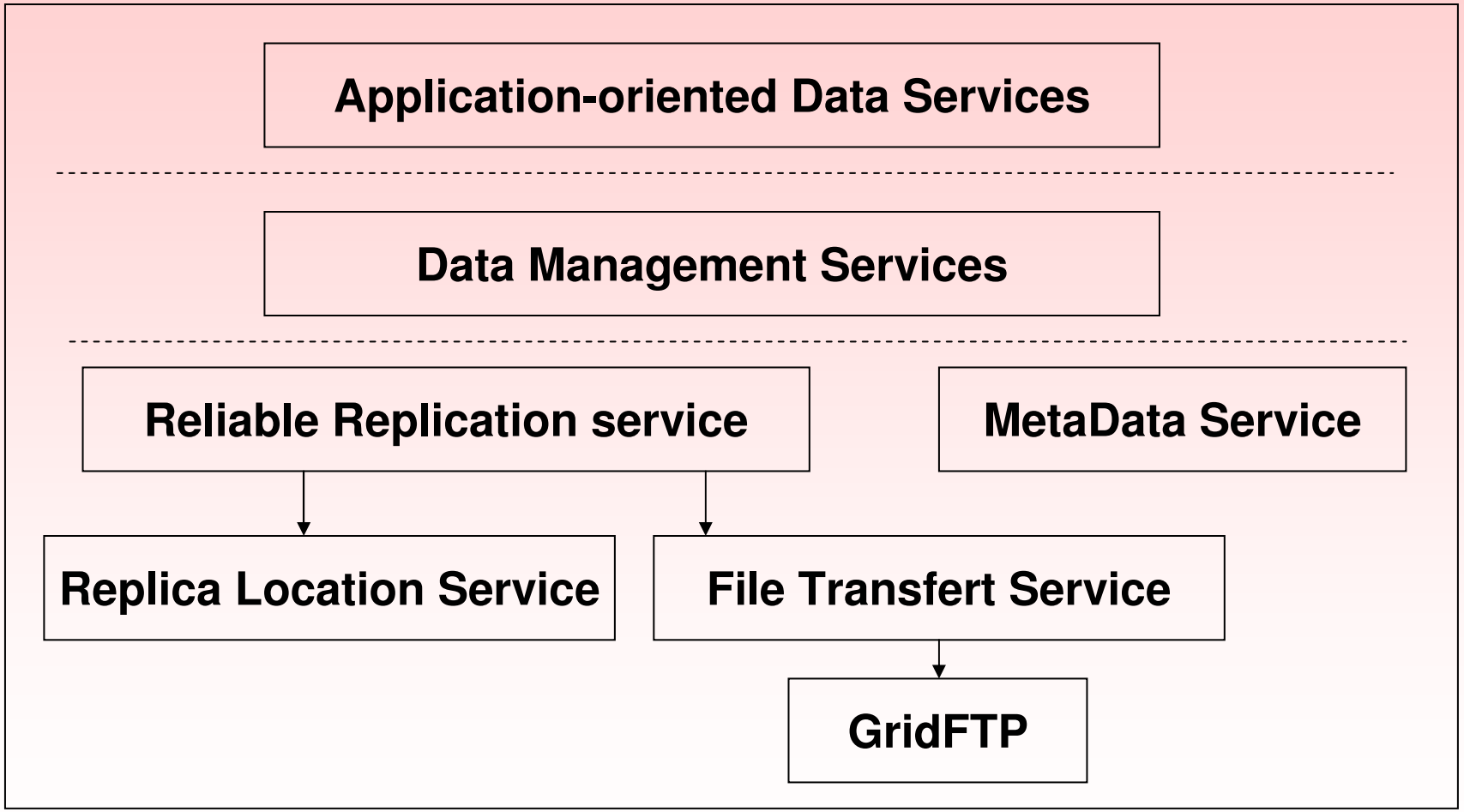


Réplique des fichiers Vs **Réplique des objets**

La Réplique des données



Replica Location Service



Architecture de grille de donnée incluant le RLS

Replica Location Service

- Données en lecture seule et en version
- Taille : centaine de site de duplication, 50 millions de fichiers logiques et 500 millions de fichiers physiques et copies
- Performance : 1000 requêtes/s et 200 MAJ/s .
  Temps de réponse de 10 ms a 5 s
- Sécurité
- Consistance
- Fiabilité

Replica Location Service

GIGGLE (GIGa scale Global Location Engine)

1. Local Replica Catalogs (LRC)
2. Replica Location Indices (RLI)
3. Mécanisme Soft State
4. Compression
5. Partitionnement et Merbership

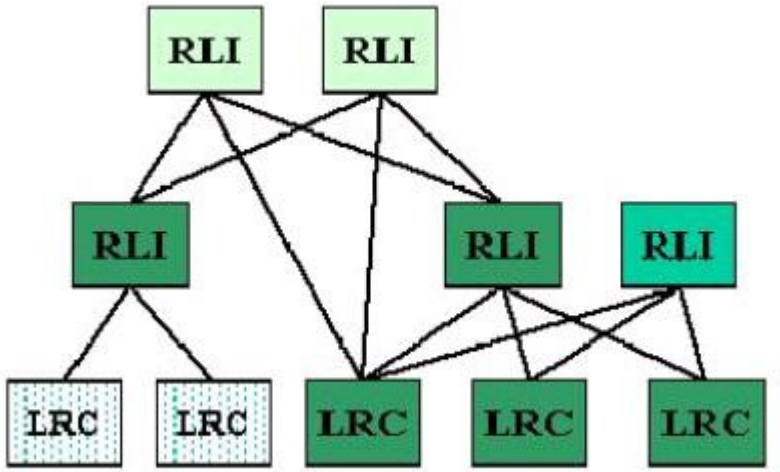
Local Replica Catalogs

- Maintenir les information sur les répliques dans un seul site de reproduction
- Maintenir les correspondances entre les nom logiques des fichiers et les noms physiques
- Coordination avec le système de stockage
- Sécurité . Authentification . Droits d'accés
- Propagation de son état
- Répondre au requêtes (LFN et PFN)

Replica Location Indice

- **Structure en Index Permettant a Satisfaire les requêtes pour plusieurs site de réplique**
- **RLI contient un ensemble d'entrée (LFN , Pointeur vers LRC)**

Topologie Hiérarchique d'un RLS



Mécanisme Soft State

- Utilisation du Soft State Protocol pour que le LRC envoie périodiquement sont état au RLI, qui incorpore les informations dans son indice (MAJ)
- Time Out : Suppression automatique des sites inaccessibles
- RLI's endommagés peuvent être restituer

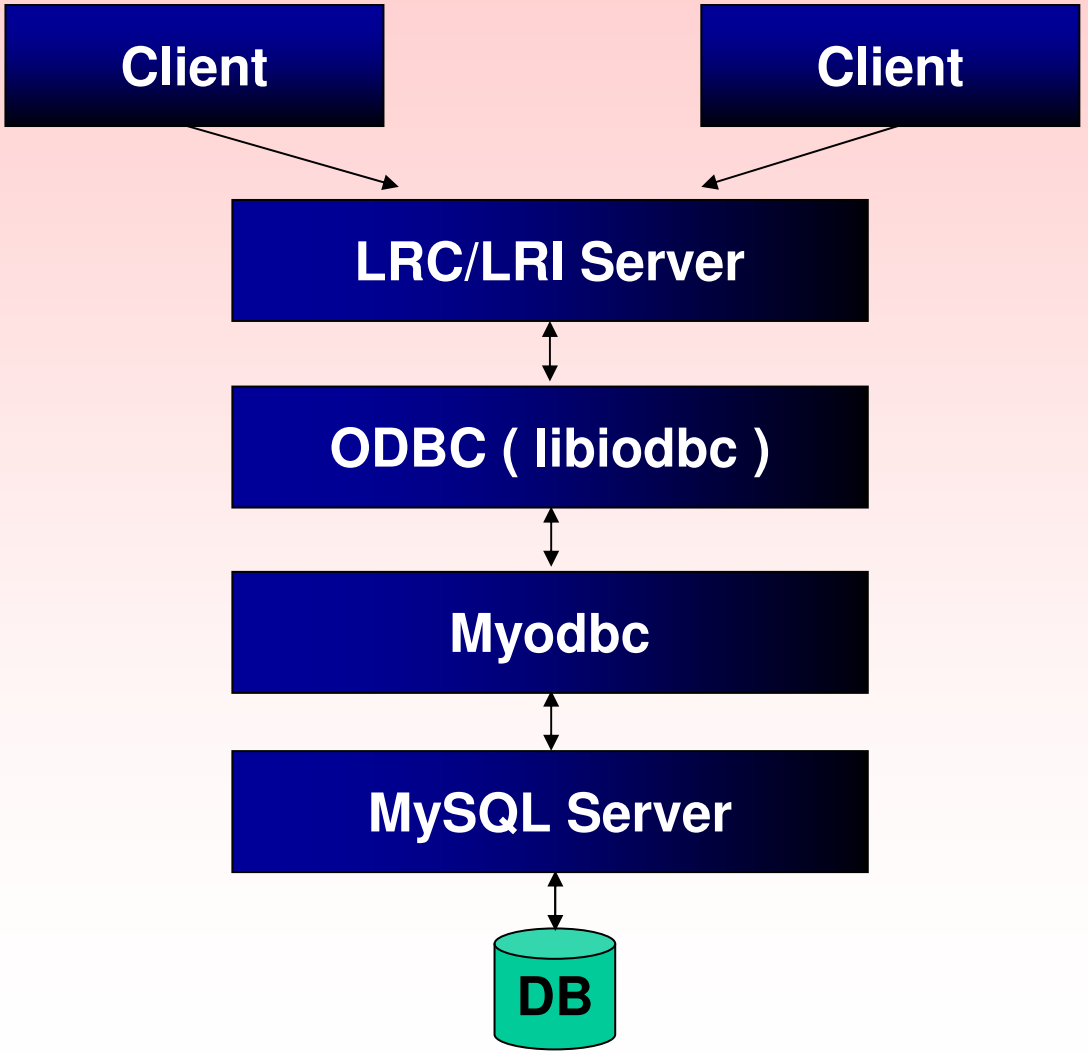
Compression

- Compression des informations du Soft State communiqué par les LRC's aux LRI's
- Réduire le trafic réseau et le coût de la maintenance des LRI's
- Utilisation des tables de Hachage comme le filtre de Bloom
- Utilisation d'information structurelle ou sémantiques des LFN (Collection)

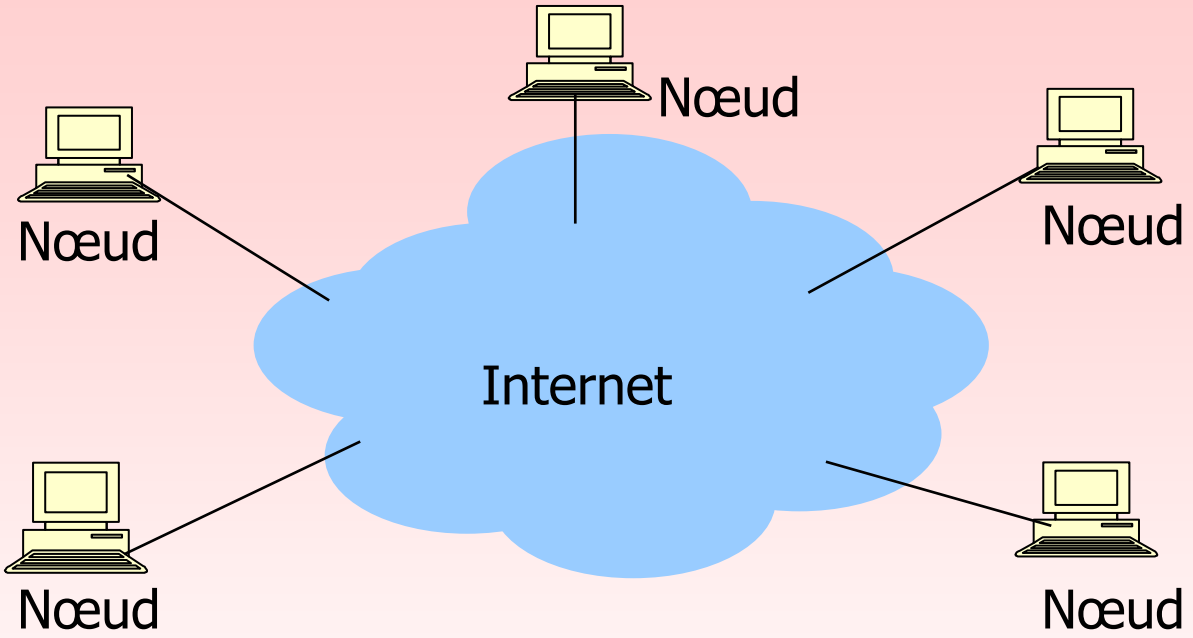
Partitionnement et Memebrship

- Partitionnement : divisé l'espace des noms logiques entre les différents RLI's pour réduire la taille du Soft State Maj
- Membership : RLI est comparé a une base de registre qui garde trace des RLI's et LRC's quand ils entrent ou quittent le système

Implémentation



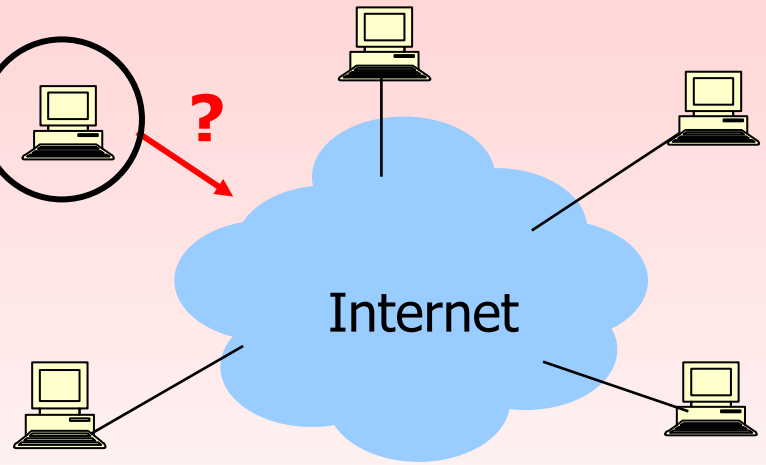
Peer to Peer



- Une architecture de système distribué :
  - Sans contrôle centralisé
  - Symétrie fonctionnelle des nœuds
- Une architecture pour la très grande échelle

Peer to Peer structuré

- **Sources des difficultés**
  - Comment obtenir des réponses pertinentes ?
  - Placement arbitraire des objets
- **Simplifier le problème**
  - Une clé unique pour chaque objet
  - Affectation « intelligente » des clés aux nœuds
  - Trouver l'objet à partir de clé
- **Solution : table de hachage Distribuée DHT**

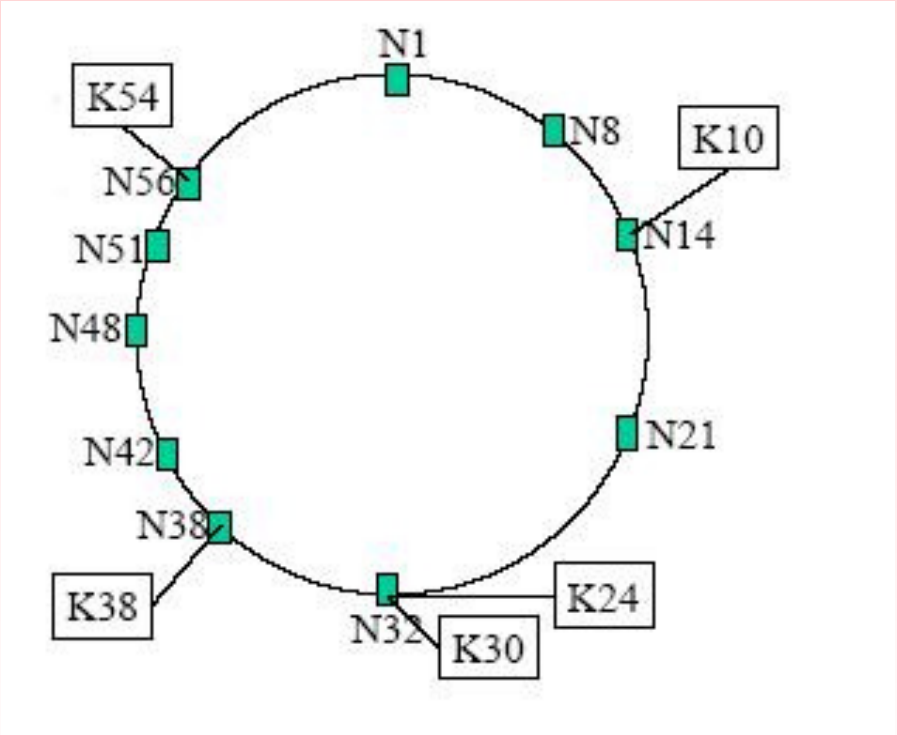


Chord

- Table de hachage distribuée
- Les noeuds sont répartis sur un anneau
- Les ressources sont réparties sur les différents noeuds de l'anneau
- Structure dynamique
  - Ajout/retrait de noeud
  - Panne d'un noeud
- Peut être utilisée pour construire des applications au-dessus (DNS, …)
- Les clés et les nœuds ont des identifiants uniques sur *m* bits

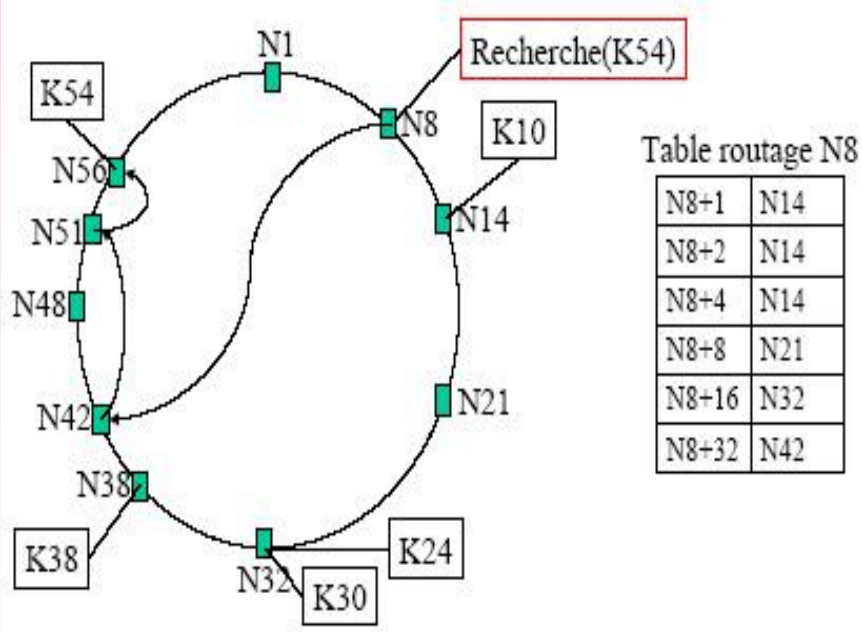
Chord

- Chaque noeud est alloué sur l'anneau en fonction de hash(IP)
- Au plus 2m noeuds
- Hash(ressource)=k
- k placé sur successeur(k) successeur(k)=noeud immédiatement supérieur (ou égal) à k



Chord

- Rechercher si la clé existe localement. Si oui on renvoie la valeur associée sinon
- Rechercher dans table routage noeud avec plus grande valeur inférieure ou égale à la clé cherchée
- Transmettre la requête au noeud sélectionné et appliquer récursivement
- Nombre de sauts moyen : O(log2(N))



Chord

- Algorithme assez simple, avec de bonnes propriétés démontrables
- Résultats expérimentaux confirment
- coût de recherche , MAJ, ajout optimaux
- Problème de latence :

– Fonction de recherche minimise le nombre de sauts, mais tous les sauts n'ont pas forcément le même prix (traversée transatlantique e.g)

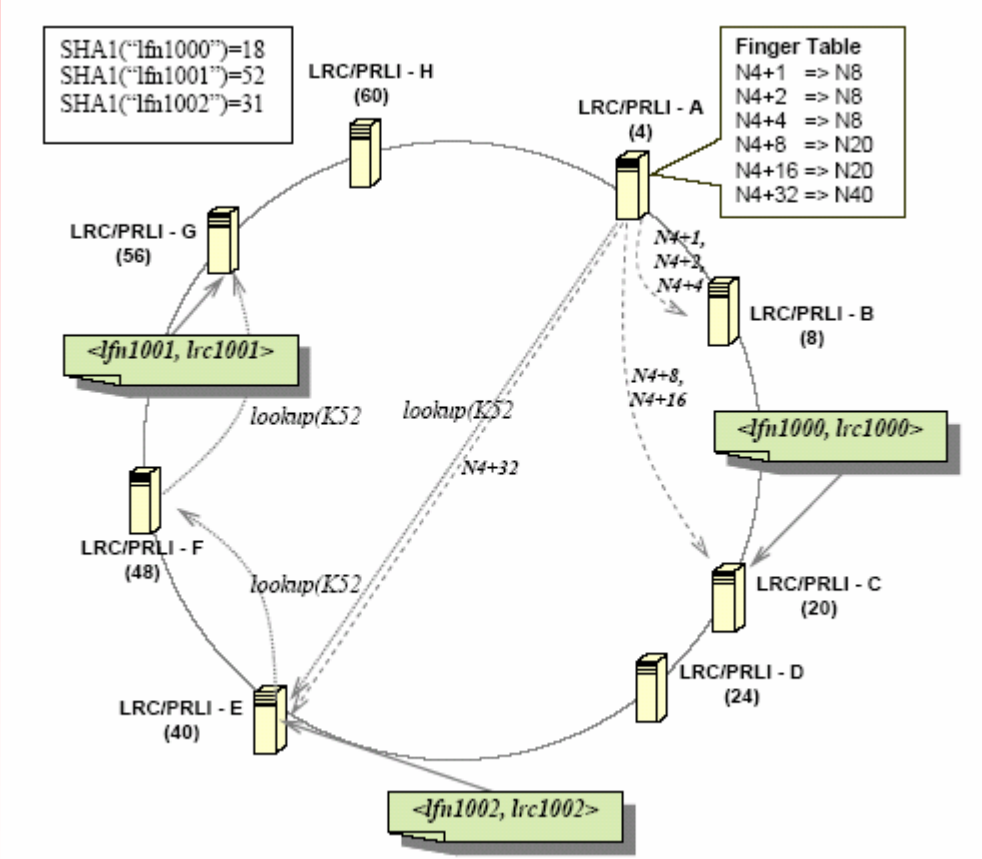
– Besoin d'utiliser de l'information sur la distance entre les noeuds (on choisira parmi les successeurs possibles celui à distance minimale) -> Global Network Positionning

Peer to Peer Replica Location Service

- Remplacement des RLI's Hiérarchiques par un réseau auto organisateur P2P de P-RLI nœuds
- LRC reste inchangé
- Chaque LRC a un P-RLI server associé a lui
- Chaque P-RLI a un m bits identificateur
- Si un P-RLI entre ou quitte le reseau , la topologie est preservé grace au stabilisateur de Chord
- Le soft State n'est pas compressé est contient la correspondance (LN , LRC)

Peer to Peer Replica Location Service

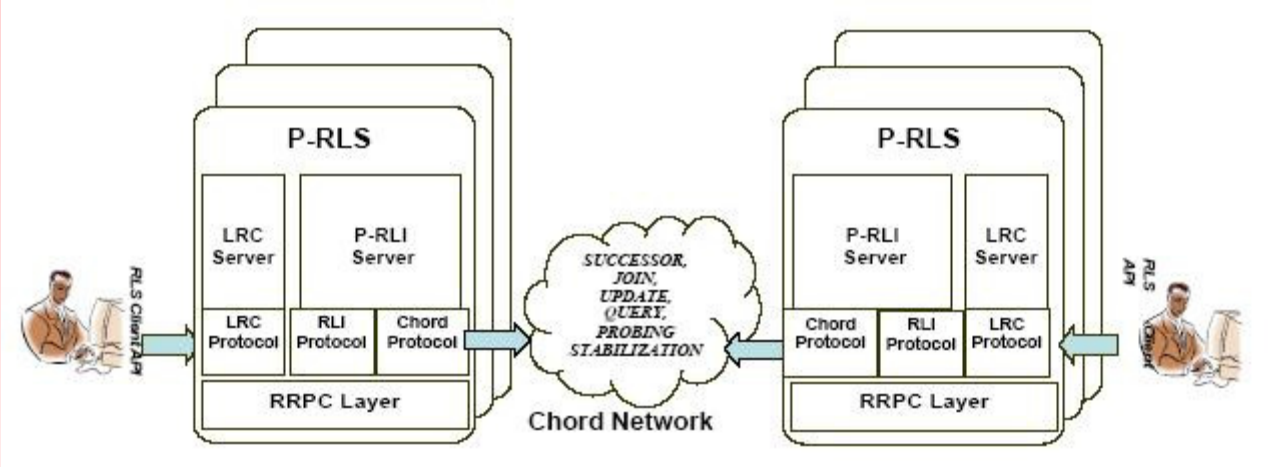
¤ Génération des clés chord des LN's en utilisant SHA 1
¤ identification du P-RLI successeur du clé chord et stockage de la correspondance (LN,LRC) dans le nœud
¤ utilisation de la méthode de recherche de Chord pour trouvé un objet



Peer to Peer Replica Location Service

- Réplique adaptative : Duplication des correspondances du nœud racine vers les k successeurs (k facteur de réplique)
- Elle permet l'équilibrage de charge et la tolérance au défaut
- Elle permet une gestion du membership et de l'auto organisation
- Une réplique adaptative des prédécesseurs permet l'équilibrage de charge des requêtes et éviter le goulot d'étranglement

P-RLS Implémentation



- RLC utilise le protocole original du RLS + protocole Chord pour MAJ des correspondances
- RLI utilise un jeu de protocole de RLS et Chord

Performance du P-PLS

- Utilisation d'un cluster de 16 nœuds (PIII 547mhz 1.5go RAM linux redhat 9 et 1Gigabit Ethernet switch pour simuler et analyser le P-RLS systeme s'étendant a 10 à 10 000 nœuds et avec 500 000 correspondance (nom logique , LRC)
- Les résultats montrent une Scalability identique au Chord
- La simulation de la réplique adaptative donne des résultats satisfaisants

Conclusion

- P-RLS est une version amélioré du RLS
- P-RLS exploite le chord P2P pour avoir une auto configuration et membership, plus grande mise en echelle (scalabilté) et une haute tolerence au defaut
- La réplique adaptative permet un plus grand équilibrage de charge et une meilleur réponse au requêtes

Merci .....

?