

Consistent Hashing

Nicolas Travers

ESILV – France

ESILV

N. Travers

Chord - Hachage Cohérent

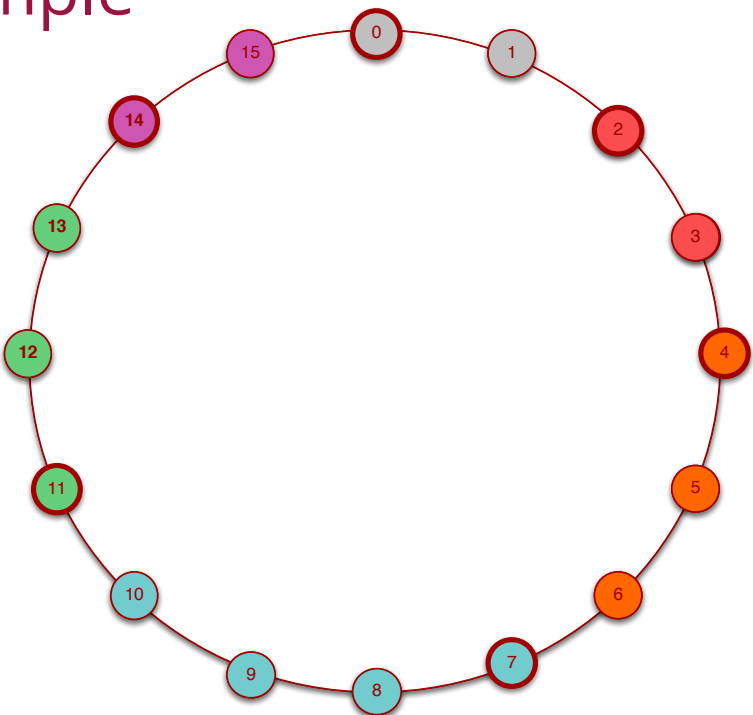
- Nombre de serveurs (virtuels) fixe
 - Puissance de 2
 - Les serveurs sont organisés en anneaux
 - Chaque serveur à en charge une partie des données de l'anneau
- Table de hachage
 - Fonction modulo du nombre de serveurs fixe
 - La table est fragmentée sur l'ensemble des serveurs

ESILV

N. Travers

Chord - Example

- 2^4 serveurs virtuels
- Serveur physique S_i :
 - Données $\rightarrow h(S_i)$
 - Données suivants $h(S_i)$ sans serveur physique
 - $h(S_i) \leftarrow h(k)$
- Soit les serveurs physiques suivants :
 - 1, 5, 8, 12, 15
 - Un nouveau serveur arrive placé en 3

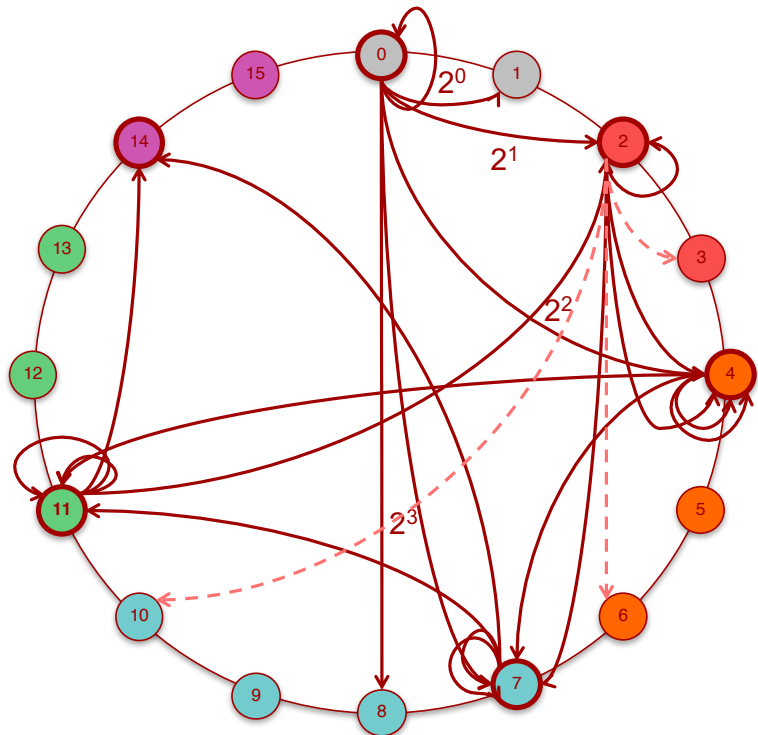


Chord - Détails

- Tolérance aux pannes
 - Réplication des données d'un serveur physique sur les 3 serveurs physiques précédents
 - Le serveur précédent a déjà les données en cas de panne
 - Un réplica peut répondre à une requête
 - Exemple : S1 est répliqué sur S3, S5 et S8
- Répartition de charge
 - Lorsqu'un serveur physique est ajouté à l'anneau, il est placé virtuellement sur 2 autres nœuds de l'anneau
 - Meilleur équilibrage (cas d'hétérogénéité des données)
 - Données répliquées encore mieux distribuées
 - Exemple : S1 est également présent en S6 et S11(16/3 ~ 5)

Chord - Table de hachage distribuée

- Chaque nœud connaît n nœuds
 - Table de hachage distribuée
 - Saut de 2^i nœuds, i entre 0 et $n-1$
 - Si $x+2^i$ est virtuel on prend le nœud qui en a la charge (précédent)



Chord - Commentaires & Algorithmme

- Table de hachage distribuée
 - Maintenance locale
 - Distribution de charge
 - Pas de contrôleur centralisé
- Algorithme de routage
 1. Recherche de v sur le serveur S_i
 2. On cherche le serveur $x = h(v)$
 3. Si i est en charge de $x \rightarrow$ Fin
 4. Dans la table de hachage locale
 - Si $T(i + 2^0) \leftarrow x \leftarrow T(i + 2^1) \rightarrow$ Serveur $T[0]$
 - Si $T(i + 2^{j-1}) \leftarrow x \leftarrow T(i + 2^j) \rightarrow$ Serveur $T[j]$
 - Si $T(i + 2^{n-1}) \leftarrow x \rightarrow$ Serveur $T[n-1]$

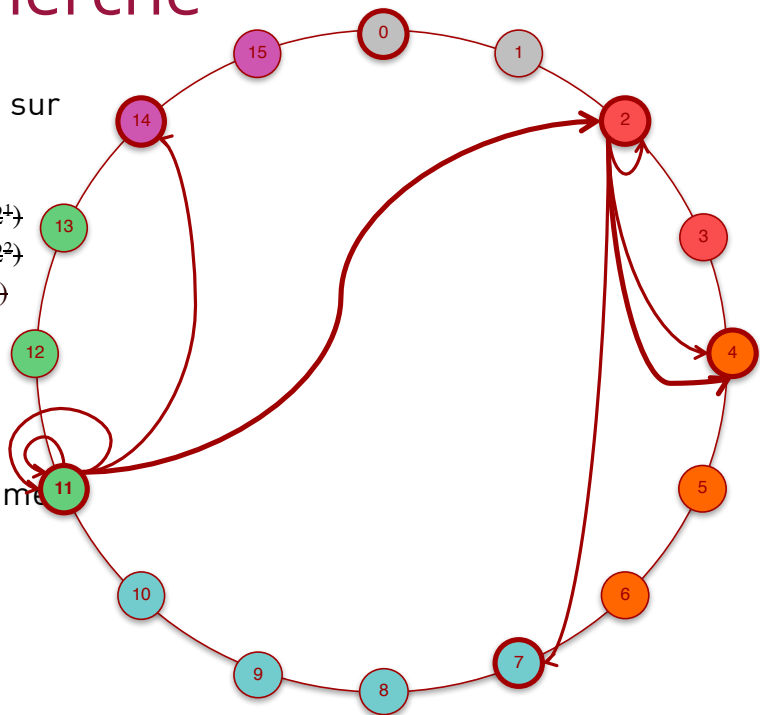
Chord - Recherche

- La valeur 54 est recherchée sur le serveur S_{11}

- $H(54) = 6$
- $T(11+2^0) = 11 \leq 6 < 11 = T(11+2^1)$
- $T(11+2^1) = 11 \leq 6 < 14 = T(11+2^2)$
- $T(11+2^2) = 14 \leq 6 < 2 = T(11+2^3)$
- $T(11+2^3) = 2 \leq 6$
- $T(2+2^0) = 2 \leq 6 < 4 = T(2+2^1)$
- $T(2+2^1) = 4 \leq 6 < 4 = T(2+2^2)$
- $T(2+2^2) = 4 \leq 6 < 7 = T(2+2^3)$

- Cout maximum de l'algorithme

- $O(\log(2^n)) \rightarrow O(n)$
- $n \rightarrow$ Nombre de nœuds
actuel



Chord – Ajout/Suppression de nœuds

- Ajout :

S_i contacte S_j , allocation dans l'anneau (nœud i)

- Table de routage :
 - S_j utilise sa table de routage pour trouver les nœuds composant la table de routage de S_i
 - Les nœuds de l'anneau doivent être mis à jour
 - Contact de $i - 2^0, i - 2^1, i - 2^2, i - 2^3$ et $i - 2^4$
 - Si absent, contacter le nœud physique suivant
- Données :
 - Récupération des données du serveur précédent : S_{i-1}
 - Valeurs v de S_{i-1} dont : $h(S_i) \leftarrow h(v)$

- Suppression :

- Les réplicas (3 prédécesseurs) garantissent la sortie d'un nœud
- Mise à jour des tables de routages

Chord - Conclusions

- Table de hachage distribuée
 - Pas de table de hachage globale
 - Efficace : $O(n)$
 - Réplication pour la tolérance aux pannes
- Utilisé pour DynamoDB (Amazon), Cassandra (Facebook)