Structures de données avancées :

*LH**

Pr ZEGOUR DJAMEL EDDINE
Ecole Supérieure d'Informatique (ESI)
www.zegour.uuuq.com
email: d_zegour@esi.dz

LH*

Présentation

- **Une SDDS basée sur Linear Hashing (LH)**
  - **LH : proposé par Litwin 1978, (VLDB-78), décrit dans plusieurs livres sur SGBDs.**
  - **LH* : Proposée par Litwin, Neimat, Schneider (ACM-Sigmod 1993)**
- **Plusieurs variantes proposées depuis par d'autres chercheurs (voir ACM-Sigmod 1994, FODO 94…)**

LH*

Rappel LH

- LH = algorithme de hachage extensible
(Etendre l'espace d'adressage primaire pour éviter l'accumulation de débordements et la détérioration progressive de performances d'accès
- Le fichier est composé de cases de capacité b >> 1
- Hachage par division h_i (c) = c mod 2^i N
- Éclatement de cases en remplaçant h_i avec h_{i+1} ; i = 0,1,..
- En moyenne, b/2 clés sont transférées vers une nouvelle case

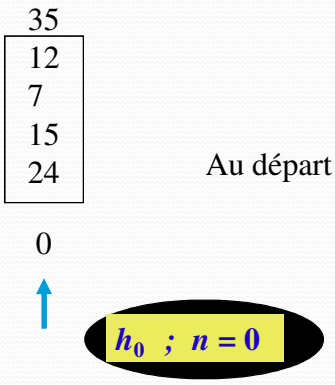
LH*

Rappel LH

- Un éclatement a lieu quand une case déborde
- On n'éclate pas la case qui déborde, mais celle pointée par un pointer *n*.
- *n* évolue : 0 /0,1/ 0,1,..,2,3/ 0,1,..,7/ 0,..,$2^i N$ / 0..
- Ces principes évitent l'existence d'un index, caractéristique d'autres algorithmes de hachage extensible.

LH*

Rappel LH

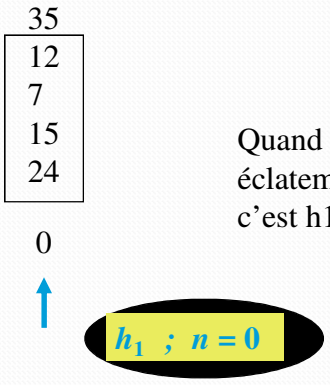


Au départ

$N = 1$
$b = 4$
$i = 0$
$h_0 : c \rightarrow 2^0$

LH*

Rappel LH

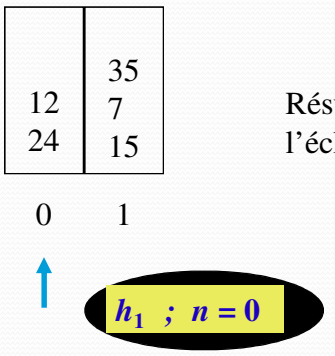


Quand il y a éclatement c'est h1

$N = 1$
$b = 4$
$i = 1$
$h_1 : c \rightarrow 2^1$

LH*

Rappel LH

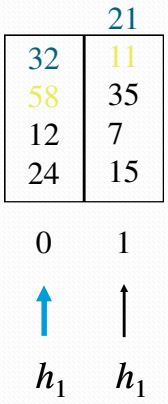


Résultat de l'éclatement

$N = 1$
$b = 4$
$i = 1$
$h_1 : c \rightarrow 2^1$

LH*

Rappel LH

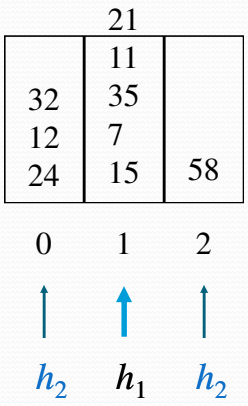


Après quelques insertions

$N = 1$
$b = 4$
$i = 1$
$h_1 : c \rightarrow 2^1$

LH*

Rappel LH

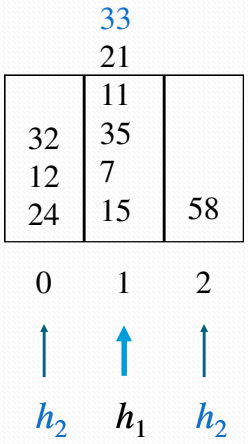


$N = 1$
$b = 4$
$i = 1$
$h_2 : c \rightarrow 2^2$

Quand il y a éclatement c'est h2

LH*

Rappel LH

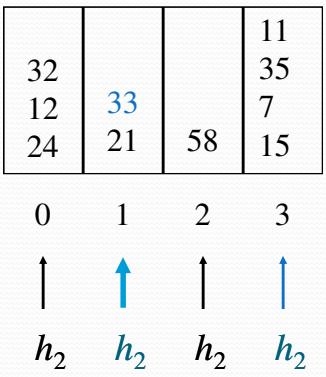


$N = 1$
$b = 4$
$i = 1$
$h_2 : c \rightarrow 2^2$

Après quelques insertions

LH*

Rappel LH

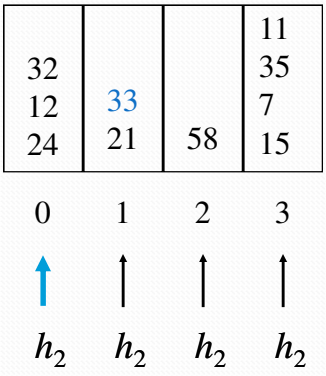


$N = 1$
$b = 4$
$i = 1$
$h_2 : c \rightarrow 2^2$

Résultat de l'éclatement

LH*

Rappel LH



$N = 1$
$b = 4$
$i = 2$
$h_2 : c \rightarrow 2^2$

N remis à zéro et i passe à 2

LH*

Rappel LH

- **Et ainsi de suite**
  On introduit h_3 puis h_4…

- **Le fichier peut s'étendre autant qu'il faut, sans jamais avoir beaucoup de débordements.**

LH*

Rappel LH

Calcul d'adresse

a ← h_i(c)
si a < P
a ← h_{i+1} (c)
Fsi

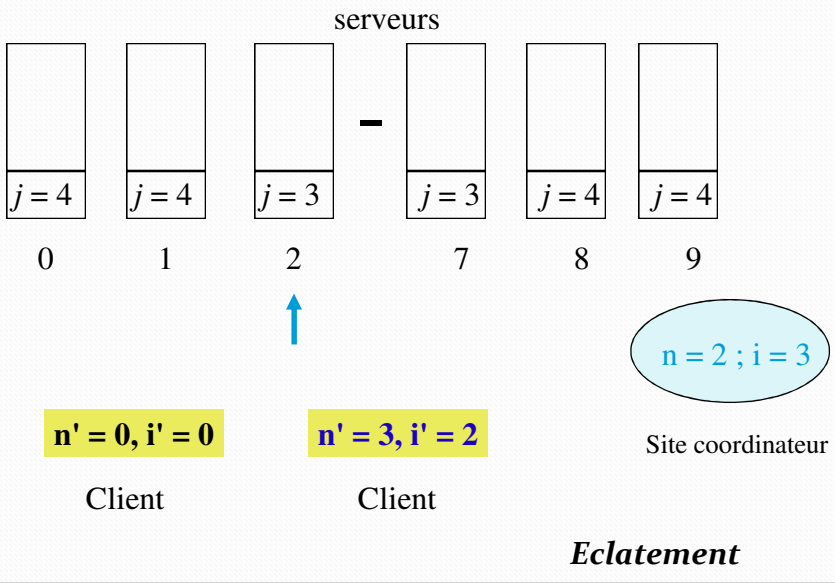
P étant la prochaine case à éclater

LH*

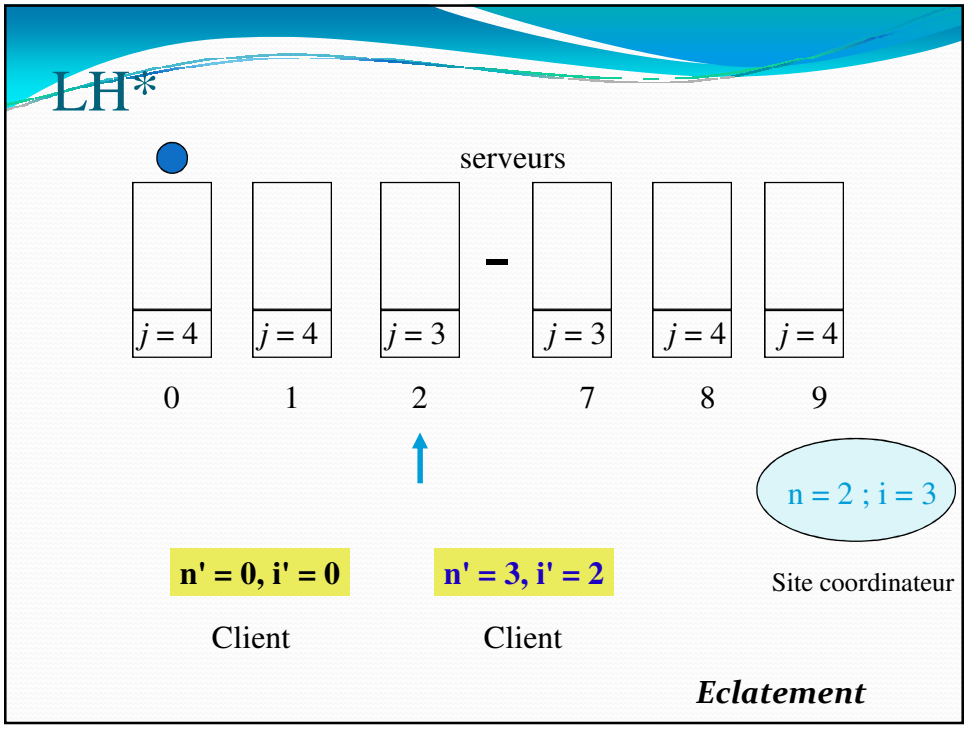
LH*

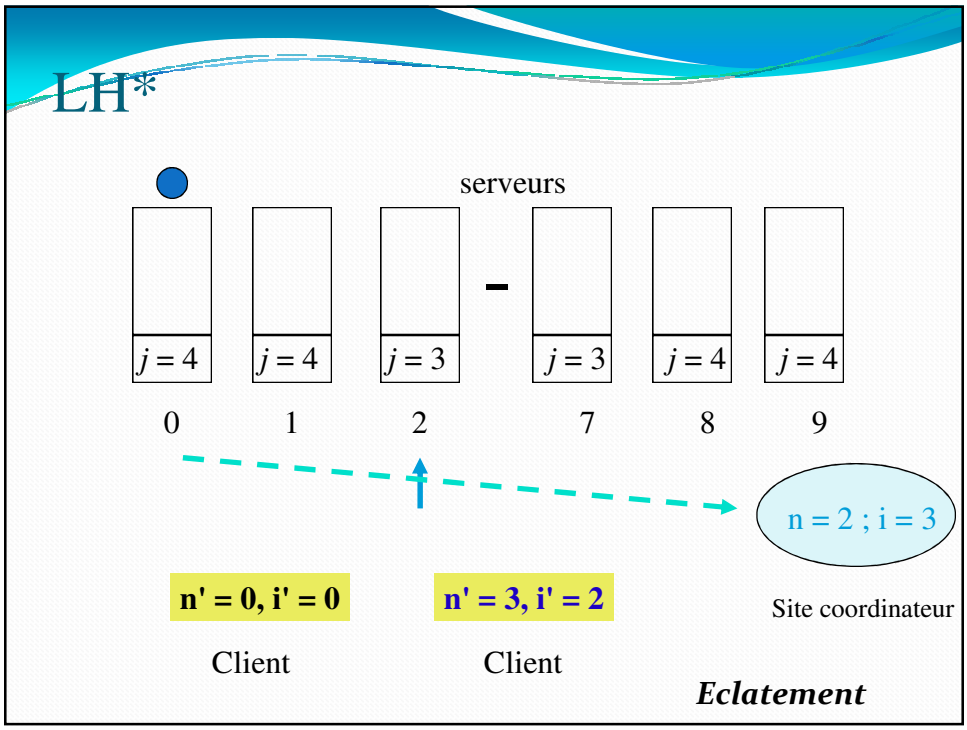
- **Propriété de LH :**
  - **Une clé *c* est dans une case *m* adressée par une fonction h_j ssi $h_j(c) = m$;**
  - ***le niveau de m est soit j = i soit j = i + 1***
- **Idée pour LH* :**
  - **mettre chaque case sur un serveur différent**
  - **Ajouter dans la case son niveau**
- **Niveau : moyen de vérifier que l'adresse d'une clé (obtenu par le client) est la bonne.**

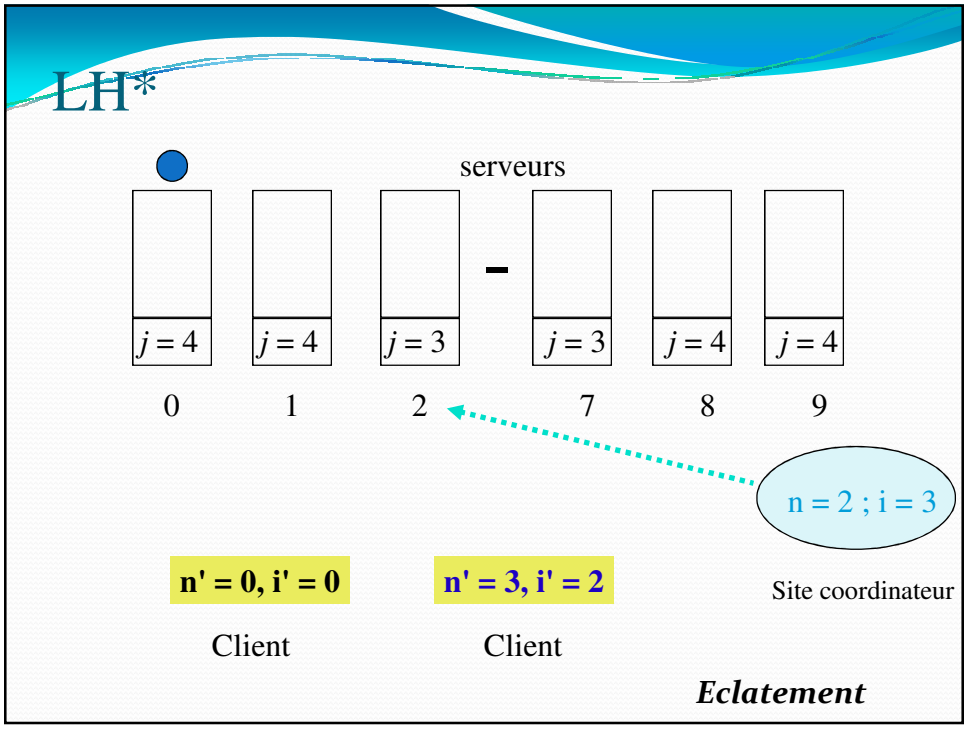
LH*

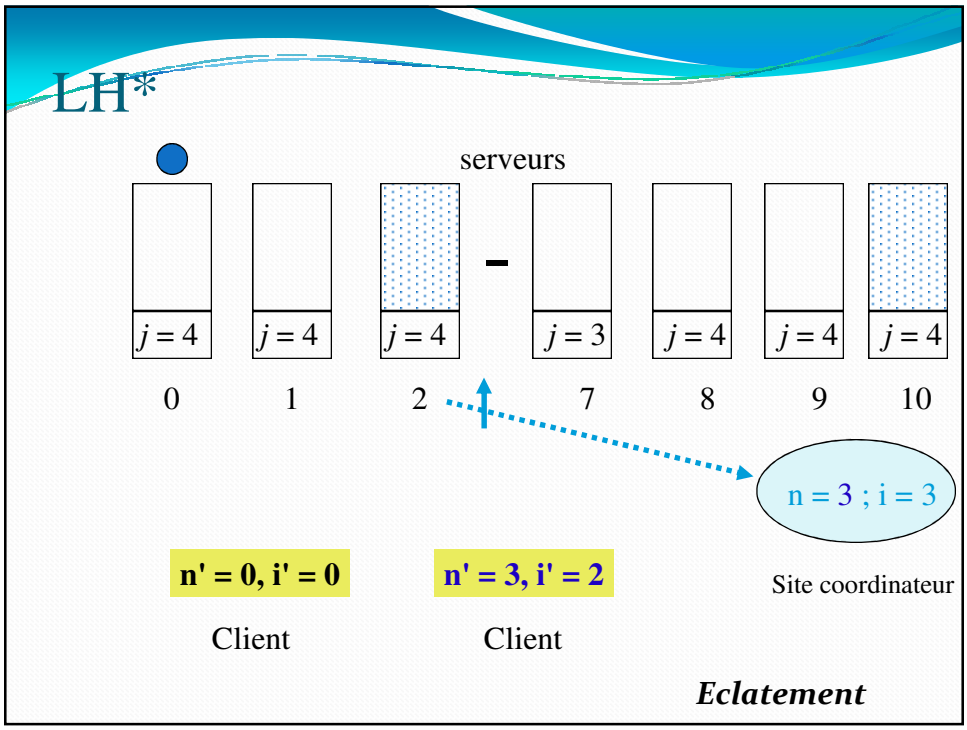


Eclatement


LH*
serveurs
j = 4
j = 4
j = 3
j = 3
j = 4
j = 4
0
1
2
7
8
9
n = 2 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Eclatement


LH*
serveurs
j = 4
j = 4
j = 3
j = 3
j = 4
j = 4
0
1
2
7
8
9
n = 2 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Eclatement


LH*
serveurs
j = 4
j = 4
j = 3
j = 3
j = 4
j = 4
0
1
2
7
8
9
n = 2 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Eclatement


LH*
serveurs
j = 4
j = 4
j = 4
j = 3
j = 4
j = 4
j = 4
0
1
2
7
8
9
10
n = 3 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Eclatement

LH*

Rôle d'un Site coordinateur

- Détermine quelle case doit être éclaté et quand.
- Problème
  - Peut tomber en panne
  - Chaque collision coûte un message et un éclatement coûte 2 messages au SC.
- Solution : utilisation d'un jeton (pour désigner la prochaine case à éclater

LH*

Utilisation d'un jeton

(Soit d la charge d'une case et b sa capacité (d > b; cas de plusieurs cases en débordement)

- Se déplace comme n (pointeur d 'éclatement)
- Quand il y a insertion dans une case qui détient le jeton, celle-ci est éclatée si d > b, puis le jeton passe à la case suivante
- Toute case qui reçoit le jeton pour la première fois s'éclate si sa charge d est supérieure à b
- → Éclatement en cascade.

LH*

Adressage

- **Le client**
  Calcule l'adresse LH de la clé *c* dans son image, soit *m*, et
  Envoie c au serveur *m*

- **Serveur *a* recevant la clé *c* (*a* = *m*) calcule :**

$a' := h_j(c)$
si $a' = a$ alors accepte *c*
sinon
$a'' := h_{j-1}(c)$
si $a < a'' < a'$ alors $a' := a''$
envoies *c* à la case a'

LH*

Proposition

L'algorithme appliqué au serveur trouve l'adresse de toute clé C envoyée par un client, et C est renvoyée au plus deux fois

La démo existe.

LH*

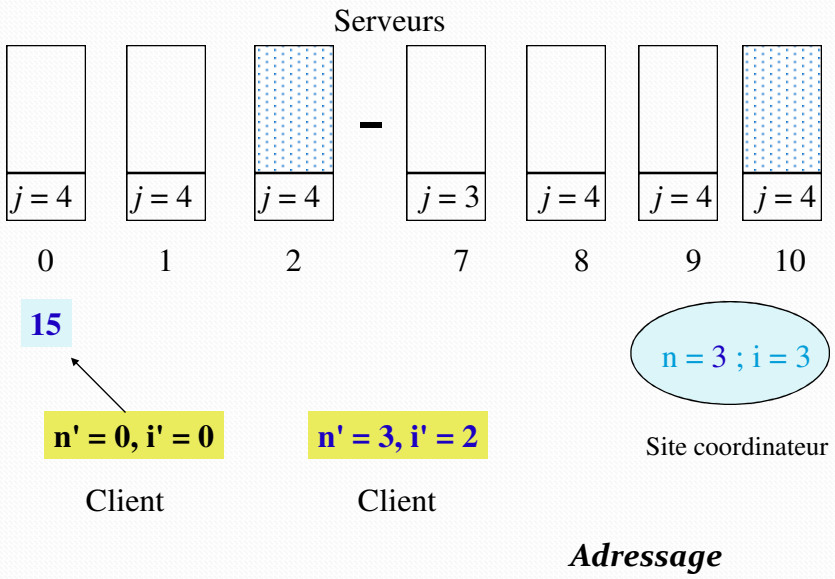
Ajustement de l'image client

- **Le message IAM consiste à ajuster l'image du client en fonction *du niveau j du serveur trouvé (directement ou après renvoi) du serveur a adressé la première fois.***

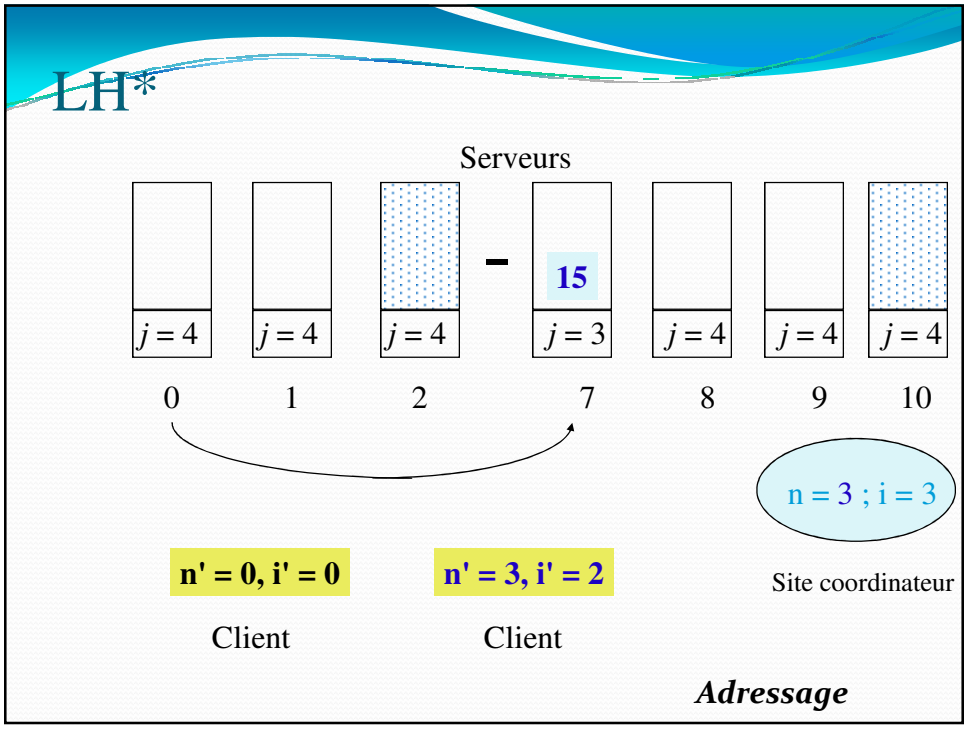
- **Soit i' *et n' les paramètres du client***

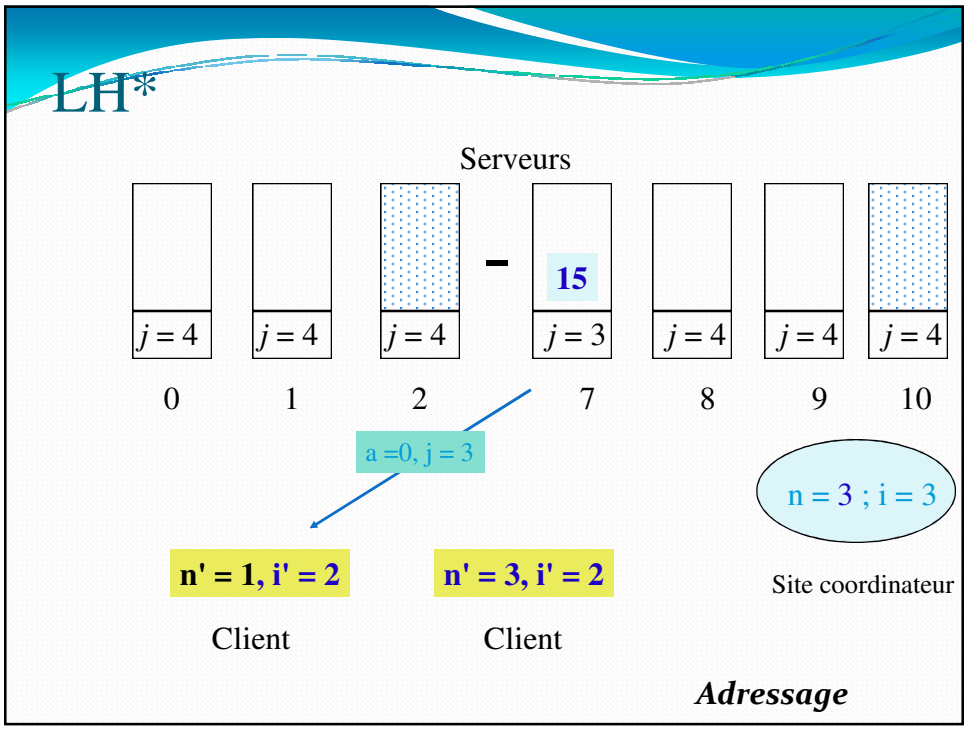
$$i' := j - 1$$
$$n' := a + 1$$
$$\text{si } n' \geq 2^{i'} \text{ alors } n' = 0,\ i' := i' + 1$$

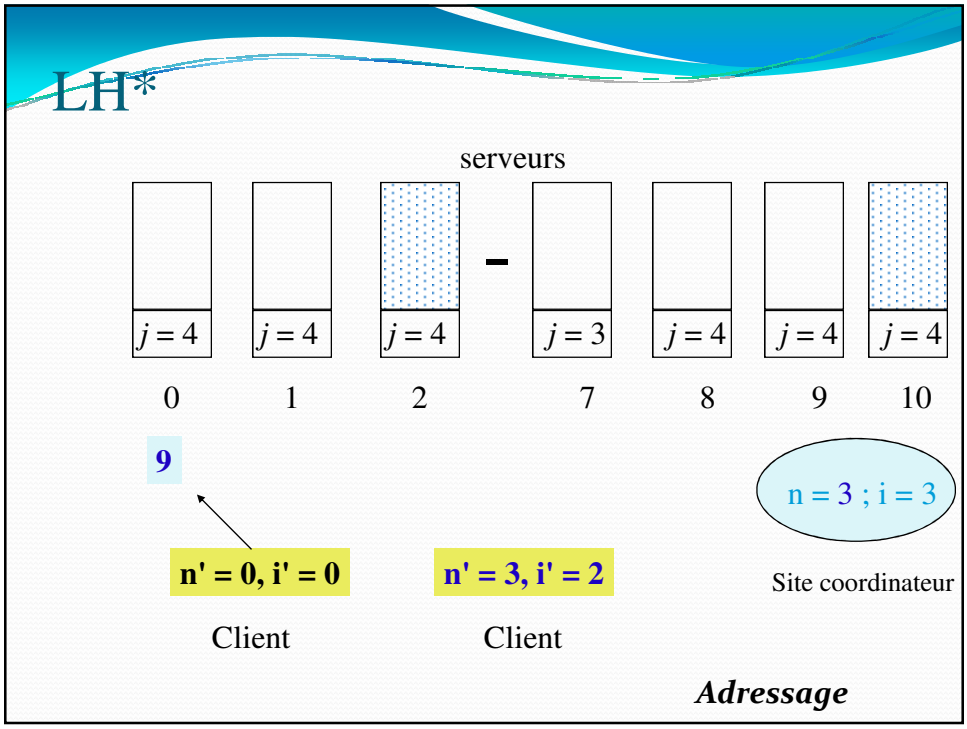
LH*

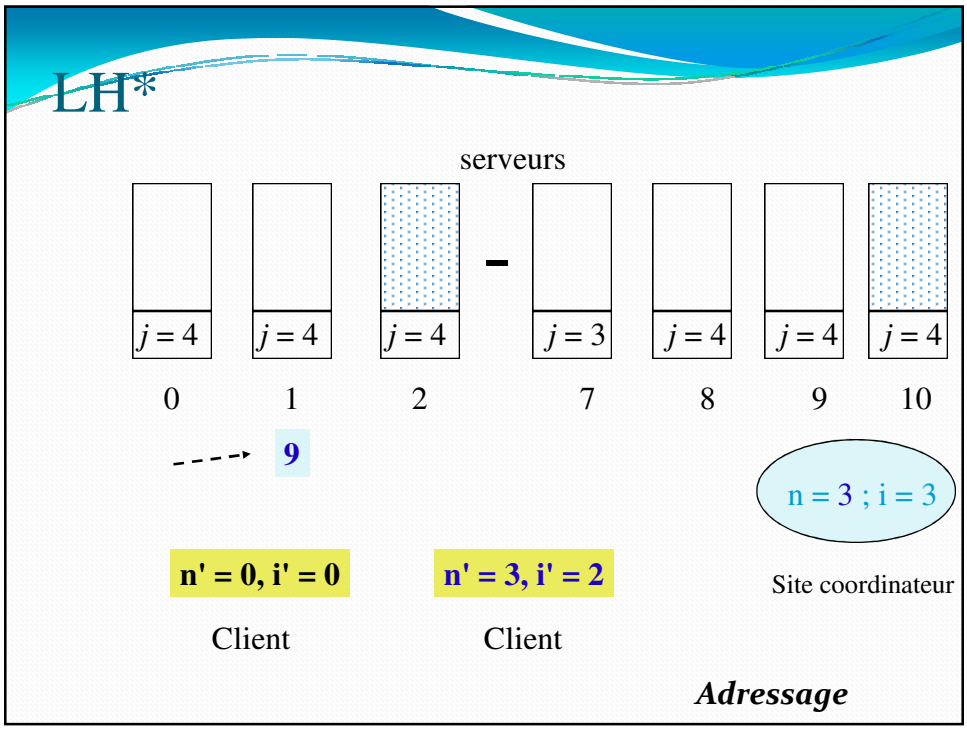


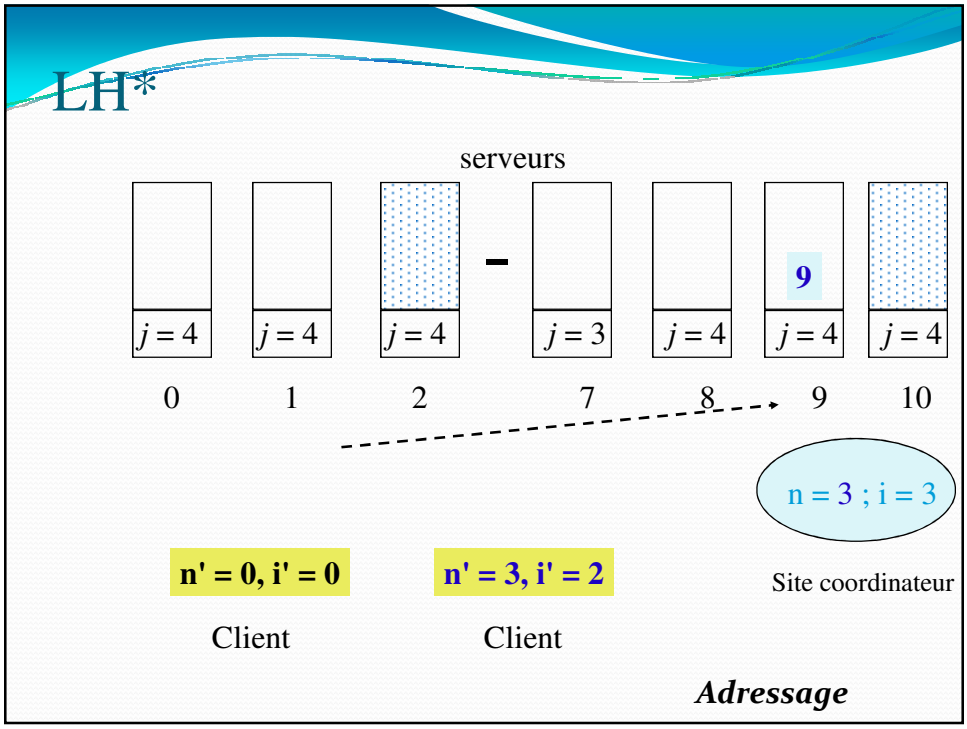
Adressage

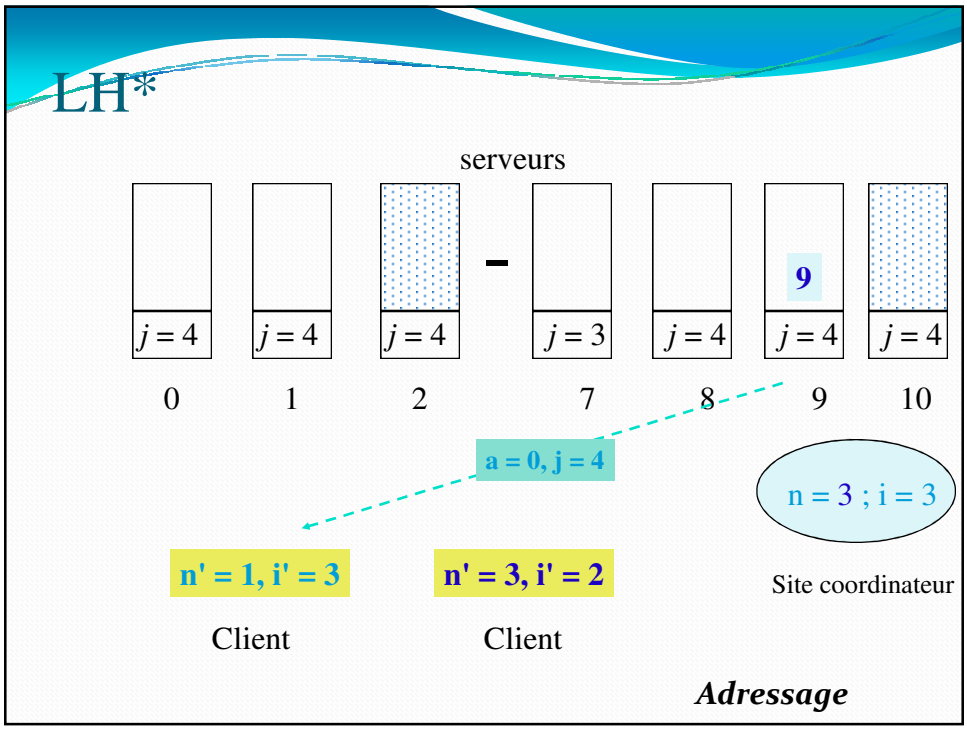

LH*
Serveurs
15
j = 4
j = 4
j = 4
j = 3
j = 4
j = 4
j = 4
0
1
2
7
8
9
10
n = 3 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage


LH*
Serveurs
15
j = 4
j = 4
j = 4
j = 3
j = 4
j = 4
j = 4
0
1
2
7
8
9
10
a =0, j = 3
n = 3 ; i = 3
n' = 1, i' = 2
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage


LH*
serveurs
j = 4
j = 4
j = 4
j = 3
j = 4
j = 4
j = 4
0
1
2
7
8
9
10
9
n = 3 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage


LH*
serveurs
j = 4
j = 4
j = 4
j = 3
j = 4
j = 4
j = 4
0
1
2
7
8
9
10
9
n = 3 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage


LH*
serveurs
j = 4
j = 4
j = 4
j = 3
j = 4
9
j = 4
j = 4
0
1
2
7
8
9
10
n = 3 ; i = 3
n' = 0, i' = 0
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage


LH*
serveurs
j = 4
j = 4
j = 4
j = 3
j = 4
9
j = 4
j = 4
0
1
2
7
8
9
10
a = 0, j = 4
n = 3 ; i = 3
n' = 1, i' = 3
n' = 3, i' = 2
Site coordinateur
Client
Client
Adressage

LH*

Requête parallèle

- ***Opération*** **envoyée par un client qui s'exécute en parallèle sur une partie du fichier (ou tout le fichier)**

(Exemple : recherche d'un ensemble d'éléments)

- **2 solutions**
  - **Utilisation des messages Broadcast / Multicast**
  - **Utilisation des messages Point à point.**

LH*

Requête parallèle

Solution Brodacast / Multicast

- **La requête Q est envoyée à tous les serveurs LH***
- **Seuls les serveurs concernés retournent des réponses au client**
- **Le client collecte toutes les réponses pour construire la réponse globale de Q.**

LH*

Requête parallèle

Solution Brodacast / Multicast

2 stratégies de terminaison:

- **Probabilistique : Un time-out est initialisé après réception de chaque article.**
- **Déterministique : Chaque serveur répond avec son numéro (case a) et son niveau j_a**

Le client termine quand il reçoit dans un ordre quelconque

$$a = 0, 1, 2, \ldots, (2^i + n + 1)$$

Avec $i = \text{Min}(j_a)$ **et**

$n = \text{Min}(a)$ **tel que** $j_a = i$.

LH*

Requête parallèle

Solution Point à point

1. Le client envoie la requête Q à tous les serveurs de son image (Q = Clé + {S1, S2, ...Sn}

2. Chaque serveur qui reçoit Q, envoie le message vers les cases ne figurant pas dans l'image du client (par propagation)

3. Seuls les serveurs concernés retournent des réponses

4. Le client collecte toutes les réponses pour construire la réponse globale de Q.

5. Même stratégies de terminaison

LH*

Requête parallèle

Solution Point à point

Algorithme de propagation (appliqué au niveau de chaque serveur a de niveau j) :

Tq j' < j
j' ← j' + 1
Envoyer (Q, j') au serveur a + $2^{j'-1}$
FTq

LH*

Résultats

On peut construire un fichier distribué grandissant à une taille quelconque et tel que :

Toute insertion et toute recherche peuvent être faites au plus en 4 messages (IAM inclus)

en général une insertion est faite en un message et une recherche en deux messages