Structures de données avancées :

Arbres B+ avec expansion partielle

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

Arbre B+ avec expansion partielle

Introduction

- **Idée : incrémenter graduellement la taille de la case surchargée au lieu de la diviser immédiatement.**
- **La division se fait quand une taille maximale est atteinte.**
- **Exige une gestion plus délicate de l'espace disque (que celle de l'arbre B) car il faut gérer des cases de tailles différentes.**
- **Intérêt : augmenter le facteur de chargement**

Arbre B+ avec expansion partielle

Concept

- **Utilise le concept d'élasticité (utilisé en 79 et poussé davantage vers les années 87).**
- **Le fichier est un ensemble de cases (bloc).**
- **Les cases sont de tailles différentes**
  Taille d'une case = x b, x=1, 2, … (nombre d'articles)
- **Incrémenter graduellement la taille de la case revient à la remplacer par une case de taille plus grande.**

Arbre B+ avec expansion partielle

Terminologie

- **Une case est étendue jusqu'à atteindre une taille maximale prédéfinie.**
- **Chaque expansion est dite <u>expansion partielle</u>**
- **Le processus qui permet à une case de pousser de sa taille minimale jusqu'à ce qu'elle se divise (taille maximale) est appelée <u>expansion complète.</u>**
- **A chaque expansion complète, une case est divisée**
- **Par conséquent, l'expansion complète incrémente le nombre de cases du fichier d'une unité.**

Arbre B+ avec expansion partielle

Terminologie

- **<u>Séquence de croissance</u> de la case :**

C'est la suite $s_0, s_1, ..., s_{r-1}$ des r différentes tailles de cases avec

$$s_0 < s_1 < ....< s_{r-1}.$$

r étant le nombre d'étapes d'expansions

- **Une séquence est <u>valide</u> que si $2s_0 > s_{r-1}$.**

(la plus grande taille est inférieure au double de la taille initiale)

Arbre B+ avec expansion partielle

Séquence la plus pratique

- **On choisit la séquence de croissance :**

$$rb, (r+1)b, ...(2r-1)b$$

- **En d'autres termes,**

  Taille minimale = rb.

  Chaque expansion partielle fait pousser la taille de b articles.

  Taille maximale = $(2r-1)b$.

Arbre B+ avec expansion partielle

Séquence la plus pratique

- Nombre maximal d'articles : $m = (2r - 1)b$
- Nombre minimum d'articles par case : $(m + 1) / 2$
- Une case de taille $s_0 = rb$ contient de $[(m+1)/2]$ à rb articles
- Une case de taille $s_i = (r+i)b$, i > o contient de $(r+i-1)b + 1$ à $(r+i)b$ articles.

  $(r+i-1)b + 1$ désigne la limite précédente + 1.

Arbre B+ avec expansion partielle

Séquence la plus pratique

Exemple :

- Posons $b=5$ avec $r=3$ expansions partielles,
- La séquence est **15, 20, 25.**
- A l'insertion du **26** ième article, la case est éclatée en deux cases de taille **15**, chacune contient **13** articles.
- Le stockage minimum pour les différentes tailles de cases est **13/15 = .866** **16/20=.8** **21/25 = .84**

Arbre B+ avec expansion partielle

Comparaison avec les arbres B*

- **Avec deux expansions partielles, le stockage minimum est de**

  67% pour les grandes cases et de

  75% pour les petites cases.

Calcul

- **r = 2 → So = 2b et S1 = 3b**
- **m = Max = 3b ; Min = (3b + 1) / 2**
- **So évolue de (3b + 1)/2 à 2b**
- **S1 évolue de (2b + 1) à 3b**
- **Minimum pour les petites cases ((3b+1)/2) / (2b) → ¾ + 1/4b → 0,75**
- **Minimum pour les grandes cases (2b+1) / (3b) → 2/3 + 1/3b → 0,67**

Arbre B+ avec expansion partielle

Comparaison avec les arbres B*

- **La valeur moyenne attendue est de 83%.**
- **Ce qui est légèrement plus haut que les 81% attendu pour les arbres B*.**

Rappel Arbre B*

✓ **Essayer de transférer les articles vers des cases (nœuds) sœurs avant de les diviser immédiatement.**

✓ **Quand deux cases sœurs sont pleines, elles sont divisée en 3.**

Arbre B+ avec expansion partielle

Apport de l'expansion partielle

- **Meilleure alternative que les arbres B* car il n'est pas nécessaire de lire les cases à gauche et à droite.**
- **Donc à priori, plus facile à implémenter et le coût de l'insertion est plus faible.**
- **<u>Inconvénient</u>**
- **La gestion de l'espace disque est plus délicate car il faut gérer des cases de tailles différentes**

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque

(une première implémentation pour r=2)

- **L'espace alloué pour le fichier est divisé en morceaux (chunks) de 5 b**
- **Quand plus d'espace est exigé pour le fichier, un morceau additionnel est demandé au système**
- **Un morceau permet de stocker 1 ou 2 cases.**
- **Chunk = unité d'allocation pour le fichier → fragmentation externe.**

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque

- **En prenant comme taille de case 2b et 3b, la classification suivante des états d'un chunk est suffisante pour notre propos :**

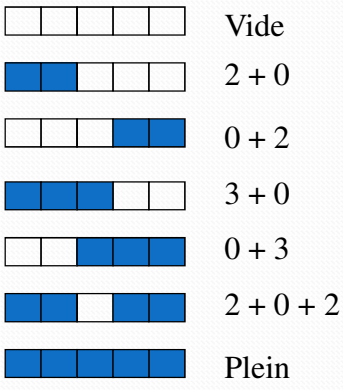
  vide, 2+0, 0+2, 3+0, 0+3, 2+0+2, plein.

- **2+0 signifie qu'une case de taille 2b occupe les deux unités les plus à gauche et le reste du chunk est libre.**

- **2+0+2 signifie qu'il y a un espace libre au milieu du chunk.**

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque



Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque
(une deuxième implémentation pour r = 2)

- **L'espace alloué pour le fichier est divisé en morceaux (chunks) de 6 b.**
- **Un morceau permet de stocker 1, 2 ou 3 cases.**
- **avec les tailles de case 2b et 3b, on a la classification suivante des états d'un chunk :**

vide, 2+0, 0+2+0, 0+2, 3+0, 0+3,
2+2+0, 2+0+2, 0+2+2, 2+3, 3+2, plein.

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque

- **Une table en mémoire est utilisée pour garder la trace des "chunks"**
- **Distinguer les différents états d'un chunk.**
- **En principe,**
  -3 bits suffisent pour les états énumérés pour les chunks de taille 5 et
  -4 bits pour les chunks de taille 6.
- **A partir de cette table, nous pouvons découvrir :**
  - ✓ **Y a t-il de la place ?**
  - ✓ **Taille exacte disponible ?**
  - ✓ **A quel endroit dans le chunk ?**

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque

- **On peut donc écrire une nouvelle case directement sans lire d'abord dans le chunk.**
- **Même pour les très grands fichiers, la table est assez petite pour être gardée en RAM.**

Arbre B+ avec expansion partielle

Une technique de gestion de l'espace disque

- **Pour les chunks de 5 b, 3 bits sont suffisant car il existe 7 états possibles**
- **Existence d'une table fictive d'association Vide : 001 ; 2 + 0 : 010 ; 0 + 2 : 011; Etc.**
- **Par programmation on traite les différents cas :**
  - ✓ **Si entrée = '001' : vide**
  - ✓ **Si entrée = '010 ': il y a de la place, pour 3 cases et à partir de la troisième**
  - ✓ **Si entrée = '111' : il y a de la place, pour 3 cases et à partir de la première**
  - ✓ **Etc.**

Arbre B+ avec expansion partielle

Algorithme d'allocation **(Cas des chunks de taille 5)**

- **Quand un espace est demandé pour une case de taille 2b,**

  un chunk contenant déjà une case de taille 2b n'est utilisé que s'il n' y a pas de chunks contenant des cases de taille 3b. Ainsi, on évite la perte de 1/5 d'un chunk.

- **Quand un espace est demandé pour une case de taille 3,**

  On essaie de le placer dans un chunk existant. Quand il n'y a pas d'espace dans un chunk, un nouveau chunk est alloué pour le fichier

Arbre B+ avec expansion partielle

Algorithme d'allocation
(Cas des chunks de taille 6)

- **Ordre de recherche pour une case de taille 2b,**

  0+2+0
  2+2+0 ou 2+0+2 ou 0+2+2
  2+0+0 ou 0+0+2
  0+3 ou 3+0

- **Ordre de recherche pour une case de taille 3b,**

  0+3 ou 3+0
  0+0+2 ou 2+0+0

Arbre B+ avec expansion partielle

Résultats :

- **Améliorer le facteur de chargement avec une gestion spéciale de l'espace disque**
- **Gain meilleur que celui de l' arbre B* (83% au lieu de 81%)**